
# WAISserver Administration Manual

**Version 2.0**
**October 1994**

A Wide Area Information Server System

WAIS Inc.'s licensor(s) make no warranties, express or implied, including without limitation the implied warranties of merchantability and fitness for a particular purpose, regarding the software. WAIS Inc.'s licensor(s) does not warrant, guarantee or make any representations regarding the use or the results of the use of the software in terms of its correctness, accuracy, reliability, currentness or otherwise. The entire risk as to the results and performance of the software is assumed by you. The exclusion of implied warranties is not permitted by some states. The above exclusion may not apply to you.
In no event will WAIS Inc's licensor(s), and their directors, officers, employees or agents (collectively WAIS Inc.'s licensor) be liable to you for any consequential, incidental or indirect damages (including damages for loss of business profits, business interruption, loss of business information, and the like) arising out of the use or inability to use the software even if WAIS Inc.'s licensor has been advised of the possibility of such damages, because some states do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitations may not apply to you. WAIS Inc.'s licensor's liability to you for actual damages from any cause whatsoever, and regardless of the form of the action (whether in contract, tort (including negligence), product liability or otherwise), will be limited to $50.



For more information about WAIS products, contact info@wais.com
For WAIS customer support, contact support@wais.com

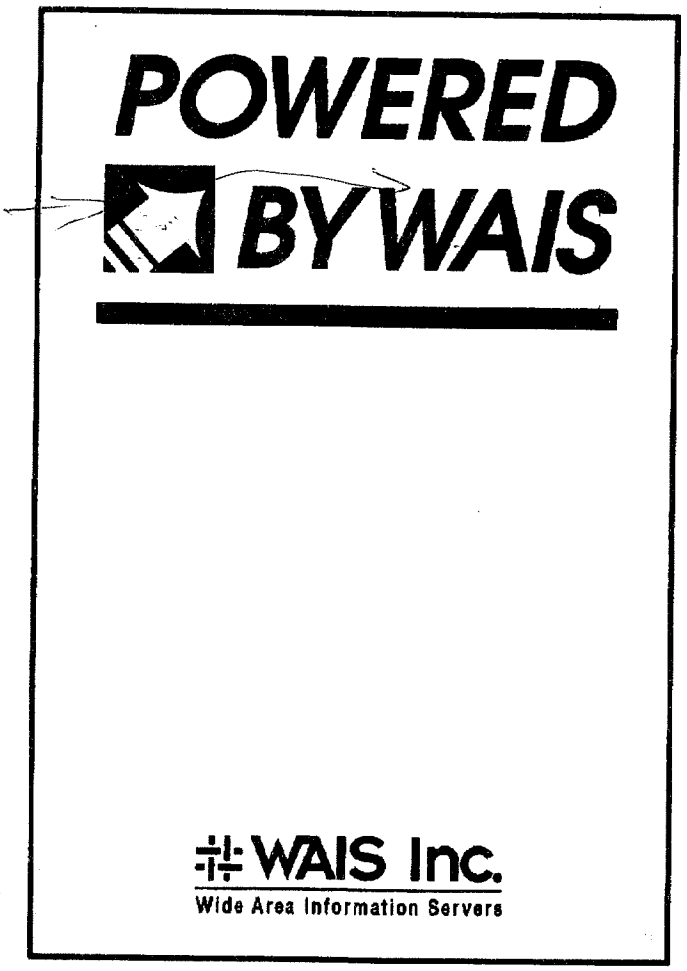
WAIS Inc.
1040 Noel Drive
Menlo Park, California 94025
(415) 327-9247

Printed in the United States of America.

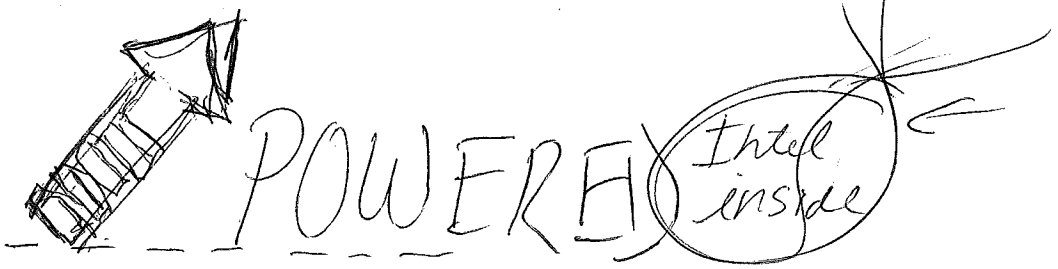
(3)

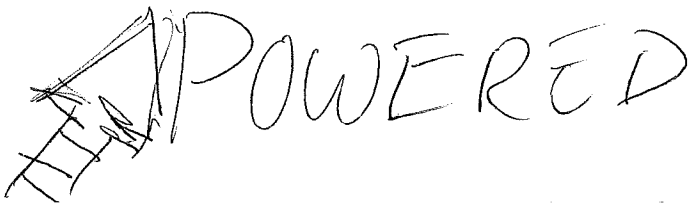
black type & arrow

red rule & box.

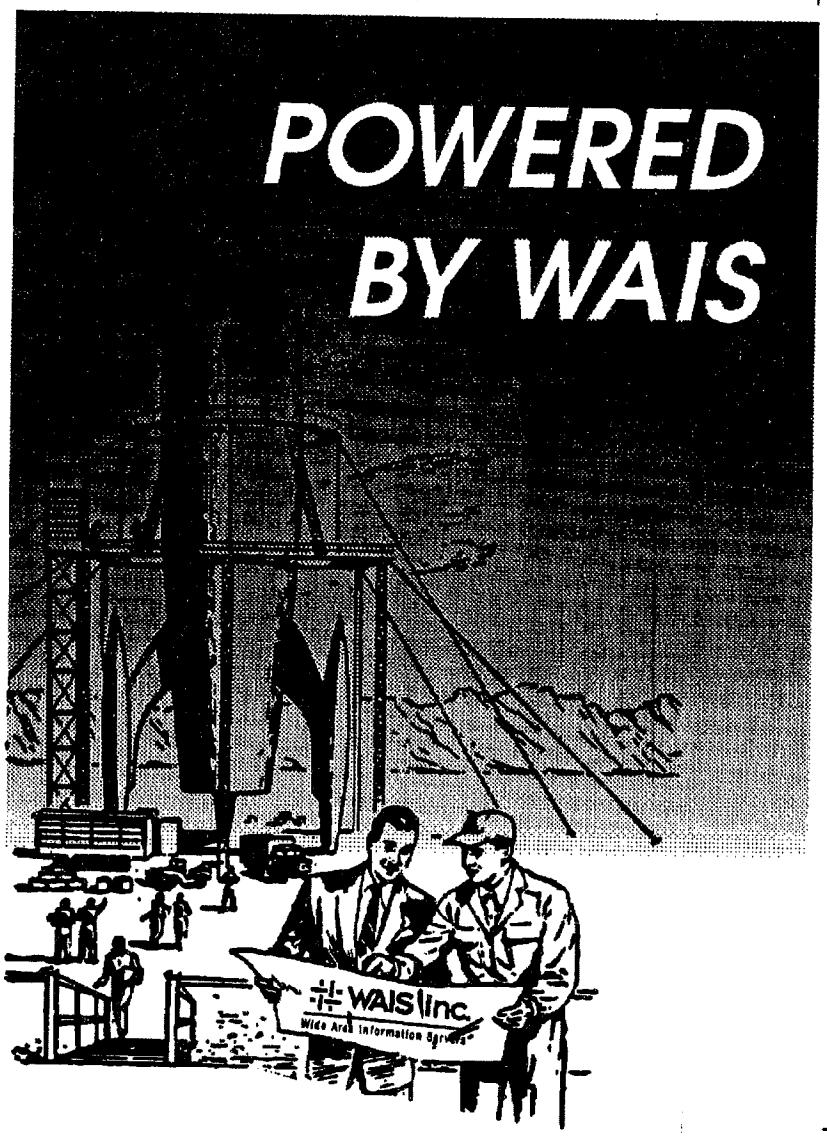



POWERE



(2) Whimsical?




rocket in black

graduation in red.

white type with black drop shadow

↑ logo on blue print

# WAIS Inc

*Wide Area Information Servers*
*1040 Noel Drive*
*Menlo Park, California 94025*
*415-617-0444*
*FAX 415-327-6513*


# FAX Cover Sheet

Date: 11/9/94

To: Sue Merrit

FAX: (212) 343-4951
Phone:

From: Lori Nelson

FAX:
Phone:

Pages Transmitted (including cover sheet):

Sue:
Here's a permission form. Please sign & return when you can. You can fax it back.

thx

Lori

WAIS Inc

*Wide Area Information Servers*

November 9, 1994

I give WAIS permission to use the Scholastic logo and home page in an Internet ad to promote the WAIS On-line Services Production Group. I have seen a comp of the ad, and approve the general concept. I understand that the artwork may change somewhat, and that the ad will debut in the CMP's Techweb service.

Name _____

Title _____

Signature _____

*1040 Noel Drive Menlo Park, CA 94025 415.327.WAIS Fax 415.327.6513 frontdesk@wais.com*

# James, Stone, Morgan & Co.

*Distributed Development & Client/Server Days™*

November 7, 1994
VIA TELECOPIER: 415-327-6513 Total Pages: 6

Lori Nelson
WAIS Inc.
1040 Noel Dr.
Menlo Park, Ca n14 94025

Dear Lori,

We are hosting a one day product showcase in your sales territory designed exclusively to **increase your short term sales.** Our one day show will put you face to face with a highly-qualified audience of hundreds of new prospects.

The show is positioned solely as a product showcase. The guests who attend will be investigating and evaluating products to develop and implement their distributed computing projects.

Your participation is easy. You can set up the morning of the show or the evening before, and be out the same day. It is a table top show, so no booths are used. It is designed to be staffed and supported with your local resources.

- Our shows effectively address the most difficult and inefficient part of the sales process, prospecting.
- Our shows are sales driven, they will beef up your sales funnels and keep you focused on closing business.
- Our one-stop shopping approach puts you in front of real specifiers and decision makers who have serious objectives and want immediate answers to their questions.
- You get the complete registration list after each show.
- Take full advantage of our Cooperative Marketing Program and you can attend for **FREE!**
- We guarantee this proven approach or your money back.

The enclosed information describes our proven approach to delivering the short-term results you need. There is still space available if you act now. We will contact you to answer any questions you may have.

Good Selling,

Larry Jamison
Director of Sales

Enclosure: eboi

5319 University Drive, Suite 500, Irvine, CA 92715-2935 Tel: (714) 854 0804 Fax: (714) 854-3396

attn: Lori Nelson

From: Judith Barker
386-3531

Following are 3 variations of sign.
I tried working:

POWERE BY
WAIS

with the logo and I can't
seem get something to look good.

Let me know what you think.

Judith

# Table of Contents

# How to Use This Manual

The *WAISserver Administration Manual* describes how to use the WAIS™ Server for UNIX®. It includes step-by-step procedures for how to set up your WAIS network publishing system.

## Who Should Read this Manual

The *WAISserver Administration* manual is intended primarily for administrators of large data sets who wish to make this data available online over a local or wide-area computer network. Some familiarity with the UNIX operating system is required.

## Organization

The *WAISserver Administration Manual* will help you build WAIS databases and help you to set up and maintain a WAIS server. The major sections are:

- *Chapter 1: Introduction* - an overview of the WAIS software suite.
- *Chapter 2: Installation* - how to set up the WAIS software.
- *Chapter 3: The WAIS Directory Structure* - A guided tour of how the databases and the server are organized.
- *Chapter 4: Building a WAIS Database* - an introduction to setting up WAIS databases.
- *Chapter 5: Setting Up a WAIS Server* - how to run a server for your databases.
- *Chapter 6: Customizing a WAIS System* - how to use additional features of the WAIS software to meet the unique needs of your database.
- *Chapter 7: Monitoring WAIS Usage* - how to keep track of your server.
- *Chapter 8: Clients and Queries* - how a WAIS database is used.
- *Chapter 9: Command Reference* - a short reference section on each of the WAIS programs.

- *Appendix A: WAIS Quick Start* - a crash course in WAIS software installation.
- *Appendix B: Recommended Reading* - a list of helpful references for UNIX administration.
- *Appendix C: Default Stopword List* - a list of the default stopwords used by the software.
- *Appendix D: WAIS Freeware Information* - Lists of unsupported freeware.
- *Appendix E : Glossary of WAIS Terms* - a list of the most frequently used WAIS terms and their definitions.

This manual will lead you through the steps of setting up and maintaining a WAIS server, and will introduce some of the considerations required in managing a server. If you are an experienced UNIX user, you may want to skip ahead to Appendix A: WAIS Quick Start. Appendix A is a crash course on getting a server up and running. Once it is set up you can return here to get a better understanding of the WAIS network publishing system.

The *WAISserverAdministration* manual is organized as an extended tutorial. If you follow it from start to finish you'll have a working WAIS server, and a good understanding of the WAISserver software. Later you can come back to it and use it as a reference manual.

# Conventions Used in this Manual

In this manual, command names and their literal arguments are printed in boldface fixed-width font, and non-literal arguments are printed in italics. For example:

**waisreporter** *log-file-name* **-d /wais/data/mydata**

Non-literal arguments, such as *log-file-name*, should be filled in with the value most appropriate for you. Lines printed by the computer are in fixed-width font:

**parsing /wais/data/jargon.231.text**

Actual commands typed at a UNIX prompt are shown preceded by the % character prompt, where the prompt character is not typed. An example usage of the **waisreporter** command might be displayed as:

**% waisreporter /wais/logs/server.log**

If an actual command requires more than a single line to display, multiple lines may be used to clearly illustrate the usage. In this case, it is assumed that the line is treated as a single command, with no intermediate carriage return characters. For example.

**% waisparse -parse text /wais/data/resumes/Smith.txt**
**-parse tiff /wais/data/resumes/Smith.tif**

is meant to be interpreted as a single command line.

# Software Release Naming Conventions

WAIS software releases are named using the following conventions:

*product-majorversion-minorversion-platform*

For example, srvr-2-0-sunos is version 2.0 of the WAISserver product for Sun computers.

At most sites, the /wais/current directory contains a symbolic link to the most recent WAISserver software release, as detailed in Chapter 2, "Installation." Therefore, throughout this manual, we will use the /wais/current directory name when giving file pathnames included in the WAISserver software release. If your installation procedures differ from those suggested in Chapter 2, substitute the appropriate directory name for your site.

# 1

# Introduction

## What is WAIS?

WAIS is a network publishing system that helps users find answers to questions from information sources over a computer network. The question or  may be expressed quite simply in natural language. Optionally, it may include a quoted literal string, use boolean syntax, or search for specific field values. The information sources may be local or remote, highly structured or free-form text. WAIS is the mechanism whereby a user can search and retrieve documents from information sources all over the world, simply by posing a question.

## WAIS Architecture

The WAIS network publishing architecture has four main components: the client, the protocol, the server and the database. The client is a user-interface program that requests services from remote or local servers. The server is a program that services these requests. The server generally runs on a machine physically connected to disks containing one or more information sources, or database. The protocol is used to connect WAIS clients and servers and is based on the NISO Z39.50 standard.

The goal of the WAIS network publishing system is to create an open architecture of information servers and clients by using a standard computer-to-computer protocol that enables users to find and question servers.

## The WAIS Software

The process of creating a WAIS database, and serving it to remote clients is illustrated in Figure 1. Following the black arrows from left to

right, the flow of information can be traced from its original repository, to its destination in response to a client's search. Taken together, the original data and the WAIS index make up a complete WAIS database. The **waislookup** command can be used for testing the database. The server executes the **waisserver** program which references the WAIS database, and returns answers to the user's question. The server also writes logs that are summarized by the **waisreporter** program.

Now consider the alternative path, from the client, asking a question of the server. The dotted arrows illustrate how a user's question and relevant documents are fed to the server. The server examines the index, which refers it to the original data. This produces a list of headlines which the server returns to the client. The same path is followed when the client makes a retrieval request. This time however, actual records from the original data are returned.

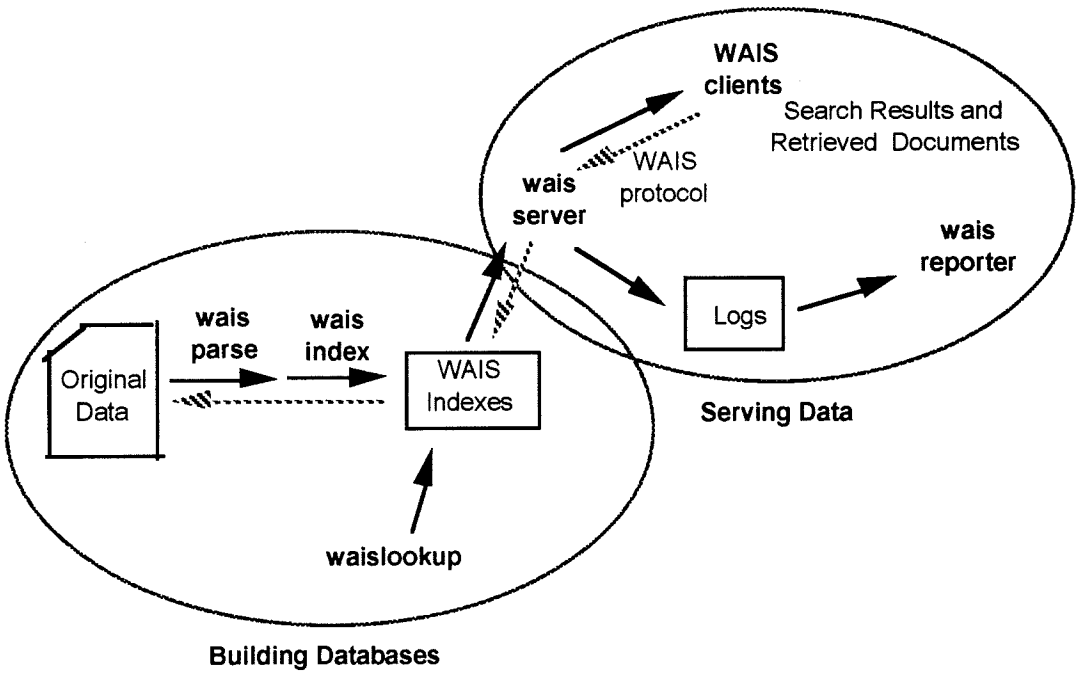



*Figure 1: Components of the WAIS System*

The important relationships are as follows:

- The **waisparse** and **waisindex** programs build a WAIS database from one or more data files which you provide.
- WAIS clients and servers communicate using the WAIS protocol.
- The **waisserver** program examines the WAIS index and the original data when it responds to a user's question.

- The WAIS database is based on the original data, and must be updated whenever that data is modified or moved.

The rest of this manual will describe how these parts fit together in the WAIS system. You will find sections on how to build and customize your WAIS database, install a server, and monitor the usage of your server.

# Getting Help

If you have problems or questions regarding the installation or use of the WAIS software, please contact us at:

WAIS support through electronic mail: support@wais.com

WAIS support hotline for WAISserver customers: 415-327-9247

WAIS Fax: 415-327-6513

Address: WAIS Inc, 1040 Noel Drive, Menlo Park, CA 94025

# Electronic Services

There are a several electronic mailing lists available to users of the WAIS software. With these mailing lists, you can receive assistance from the WAIS Inc technical support staff, and the from the experienced users of the WAIS software.

**support@wais.com**: Prompt online technical support for the WAIS Inc product line. WAIS Inc actively encourages administrators to use this list to seek help on any issue related to the WAIS Inc products. This mailing list welcomes suggestions, comments, questions, and bug reports.

**wais-talk@wais.com**: An interactive list for WAIS software developers wishing to share their insights and questions with other developers. Requests for inclusion to this mailing list should be made to wais-talk-request@wais.com.

**comp.infosystems.wais**: A network news discussion group on WAIS issues. All postings to wais-discussion@wais.com go to this group as well.

# 2

# Installation

This chapter describes the contents of a WAIS software release, and shows you how to set up WAIS software on your machine.

## Getting Started

Installing WAIS software is fairly easy, but there are a few considerations you must keep in mind before you get started. This chapter identifies them, and you can use it as a worksheet as you do your installation. Don't worry if you don't know the answer to some of the questions. You can revisit this section once you've read more of the manual.

Before you start, make sure that you have the following:

- A UNIX machine running an operating system supported by WAIS.
- Disk space of approximately 5 megabytes in which to load the WAIS software.
- Basic knowledge of UNIX. Appendix B includes references on UNIX administration that you may find useful.
- An account on a UNIX server machine.
- Superuser access to the UNIX server machine. This is only required for running the server in daemon mode. For more information, see Chapter 5, Setting up a WAIS Server.
- A TCP/IP network to connect WAIS clients.

During the installation, you'll need to make the following decisions. The questions will be fully explained in following sections. They are listed here to help you plan your installation.

- Where should the software be installed?
- What data would work best in your WAIS database?
- Which machine should you use as your WAIS server?

- What TCP/IP port should the server run on?
- Who should have access to the databases?

# Checking the Contents

Before you begin the software installation process, make sure you have a full WAIS release. It should contain the following documentation:

- *WAISserver Administration Manual* (this manual)

The software package itself is released in one of these forms:

- If you've **ftp**'d the software from WAIS Inc's **ftp** site, it will be in a compressed archive file on your hard disk. The name will be something like *product-majorversion-minorversion*.tar.Z.
- If you've received a tape from us, the software will be in an archive on the tape.

If any of these components are missing, please contact us at your earliest convenience.

# Unpacking Your WAIS System

The first step is getting the WAIS software onto your UNIX machine. At this point you'll need to decide which machine should become the WAIS server, and which directory (and disk) should hold the software. There are several considerations in deciding which machine to use as the server.

- The machine's network location should match your distribution plans. If you intend to publicize your WAIS servers to the Internet community, the machine should be directly on the Internet. If you are setting your server up for use as a confidential resource, you may want to put it on a network that is protected from the Internet by a firewall machine.
- The machine's network location and name should be fairly stable, since hundreds or thousands of clients may come to depend on the server running on the machine.
- We highly recommend that the WAIS databases be located on disks mounted directly (locally) on the server machine. The WAISserver software can use databases on disks remotely mounted through NFS, but performance is greatly reduced.

- The data which is indexed by WAIS should also be locally mounted, although this is less of a concern than the location of the WAIS index files.
- The class of machine and amount of memory should be appropriate for the server task it is to perform. Requirements are highly variable depending on the number of users and the size and character of the database. As a general rule, server machines should have at least 16 Megabytes of RAM. If the server is used lightly, other processes can easily share the machine.

Once you've decided on a machine to run the WAIS server, you'll need to decide where to put the software release. Our recommended directory layout is described in detail in the next chapter. The directory you're building now will be the root of the WAIS software tree. Here are the criterion for making your decision.

- The release itself requires about 5 megabytes. You may want to leave some spare room on the release disk for future WAIS releases and upgrades. You don't need to have room for all your databases on a single disk.
- The name of the release directory should be fairly stable, as clients will come to depend on it.

In our examples we'll use /wais as the WAIS directory. Whenever you see /wais, substitute the name you've chosen.

Once you have made your decision, create the directory, and move into it. If it already exists you won't need to do the **mkdir**'s again.

```
% mkdir /wais
% cd /wais
```

If you've **ftp**'d the software over the Internet, you'll need to uncompress it and unpack it. The file is compressed if its filename ends in .Z, and it is an archive if it has a .tar extension. The UNIX **zcat** program uncompresses the file, and passes the results on to **tar**, the UNIX Tape ARchive utility. Here's the line that does the work:

```
% mv wais-majorversion-minorversion.tar.Z /wais
% cd /wais
% zcat wais-majorversion-minorversion.tar.Z | tar -xvf -
```

If the software is on a tape, you'll have to copy the tape onto your hard disk. That means you will need to know which UNIX device describes the tape drive on your system. Your system administrator should be able to help with this. We will use a typical one in this example; it is the first tape drive on the system, called /dev/rst0 on SunOS 4.1.x, and

/dev/rdsk/c0t4d0s2 on SunOS 5.1. The WAIS software tape is not compressed, so all you need to do is **tar**.

```
% cd /wais
% tar -xvf /dev/rst0
```

In either case, the computer should print a list of the contents of the WAIS release. It will look something like this, although it will vary depending on the exact release you have received.

```
% zcat srvr-2-0-sunos.tar.Z | tar -xvf -
x srvr-2-0-sunos/bin/waisparse, 270336 bytes, 528 tape blocks
x srvr-2-0-sunos/bin/waisindex, 720896 bytes, 1408 tape blocks
x srvr-2-0-sunos/bin/waisdelete, 712704 bytes, 1392 tape blocks
x srvr-2-0-sunos/bin/waisserver, 1474560 bytes, 2880 tape blocks
x srvr-2-0-sunos/bin/waislookup, 1269760 bytes, 2480 tape blocks
x srvr-2-0-sunos/bin/waisreporter, 172032 bytes, 336 tape blocks
x srvr-2-0-sunos/bin/README, 127 bytes, 1 tape blocks
x srvr-2-0-sunos/util/update-lcl-dir-of-servers, 1024 bytes, 2 tape blocks
x srvr-2-0-sunos/util/run-waisreporter, 1947 bytes, 4 tape blocks
x srvr-2-0-sunos/util/stopwords.txt, 2002 bytes, 4 tape blocks
x srvr-2-0-sunos/util/filter-codes.h, 1544 bytes, 4 tape blocks
x srvr-2-0-sunos/util/README, 218 bytes, 1 tape blocks
x srvr-2-0-sunos/util/how-to-search, 15368 bytes, 31 tape blocks
x srvr-2-0-sunos/doc/waisparse.txt, 10432 bytes, 21 tape blocks
x srvr-2-0-sunos/doc/waisindex.txt, 8750 bytes, 18 tape blocks
x srvr-2-0-sunos/doc/waisdelete.txt, 2243 bytes, 5 tape blocks
x srvr-2-0-sunos/doc/waisserver.txt, 5697 bytes, 12 tape blocks
x srvr-2-0-sunos/doc/waislookup.txt, 5580 bytes, 11 tape blocks
x srvr-2-0-sunos/doc/waisreporter.txt, 2281 bytes, 5 tape blocks
x srvr-2-0-sunos/doc/access-lists.txt, 1726 bytes, 4 tape blocks
x srvr-2-0-sunos/doc/ethics.txt, 9338 bytes, 19 tape blocks
x srvr-2-0-sunos/doc/Release-Notes-prelim.msword, 27648 bytes, 54 tape blocks
x srvr-2-0-sunos/doc/Admin-Manual-2.0-prelim.msword, 318464 bytes, 622 tape blocks
x srvr-2-0-sunos/man/waisparse.l, 8180 bytes, 16 tape blocks
x srvr-2-0-sunos/man/waisindex.l, 6475 bytes, 13 tape blocks
x srvr-2-0-sunos/man/waisdelete.l, 1679 bytes, 4 tape blocks
x srvr-2-0-sunos/man/waisserver.l, 4414 bytes, 9 tape blocks
x srvr-2-0-sunos/man/waislookup.l, 4004 bytes, 8 tape blocks
x srvr-2-0-sunos/man/waisreporter.l, 1700 bytes, 4 tape blocks
x srvr-2-0-sunos/sample/README, 3335 bytes, 7 tape blocks
x srvr-2-0-sunos/sample/data/resumes/res3.txt, 4102 bytes, 9 tape blocks
x srvr-2-0-sunos/sample/data/resumes/README, 5243 bytes, 11 tape blocks
x srvr-2-0-sunos/sample/data/resumes/res1.txt, 3070 bytes, 6 tape blocks
x srvr-2-0-sunos/sample/data/resumes/res2.txt, 4634 bytes, 10 tape blocks
x srvr-2-0-sunos/sample/data/mail/rmail, 5760 bytes, 12 tape blocks
x srvr-2-0-sunos/sample/data/mail/README, 4394 bytes, 9 tape blocks
x srvr-2-0-sunos/sample/data/jargon/README, 4292 bytes, 9 tape blocks
x srvr-2-0-sunos/sample/data/jargon/convert.c, 1874 bytes, 4 tape blocks
x srvr-2-0-sunos/sample/data/jargon/jargon.231.text, 558266 bytes, 1091 tape blocks
x srvr-2-0-sunos/sample/data/weather/ACUS1.DOC, 4961 bytes, 10 tape blocks
x srvr-2-0-sunos/sample/data/weather/README, 2866 bytes, 6 tape blocks
```

x srvr-2-0-sunos/sample/data/weather/WXKEY.DOC, 31 bytes, 1 tape blocks
x srvr-2-0-sunos/sample/data/weather/WXMAPARS.DOC, 36 bytes, 1 tape blocks
x srvr-2-0-sunos/sample/data/weather/ACUS1.GIF, 13581 bytes, 27 tape blocks
x srvr-2-0-sunos/sample/data/weather/SELSLOG.GIF, 13870 bytes, 28 tape blocks
x srvr-2-0-sunos/sample/data/weather/WXKEY.GIF, 18865 bytes, 37 tape blocks
x srvr-2-0-sunos/sample/data/weather/WXMAPARS.GIF, 35661 bytes, 70 tape blocks
x srvr-2-0-sunos/sample/data/weather/SELSLOG.DOC, 5216 bytes, 11 tape blocks
x srvr-2-0-sunos/sample/Makefile, 1327 bytes, 3 tape blocks
x srvr-2-0-sunos/sample/currency-exchange/customcode/waisparse, 253952 bytes, 496 tape blocks
x srvr-2-0-sunos/sample/currency-exchange/customcode/localParsers.c, 13725 bytes, 27 tape blocks
x srvr-2-0-sunos/sample/currency-exchange/customcode/currency-filter, 1886 bytes, 4 tape blocks
x srvr-2-0-sunos/sample/currency-exchange/customcode/ExchangeDump.exe, 29 bytes, 1 tape blocks
x srvr-2-0-sunos/sample/currency-exchange/customcode/currency-filter.wc, 75 bytes, 1 tape blocks
x srvr-2-0-sunos/sample/currency-exchange/money.txt, 584 bytes, 2 tape blocks
x srvr-2-0-sunos/sample/currency-exchange/Tutorial, 9545 bytes, 19 tape blocks
x srvr-2-0-sunos/waisgate/INSTALL, 3595 bytes, 8 tape blocks
x srvr-2-0-sunos/waisgate/README, 5981 bytes, 12 tape blocks
x srvr-2-0-sunos/waisgate/icons/back.xbm, 506 bytes, 1 tape blocks
x srvr-2-0-sunos/waisgate/icons/ball.xbm, 437 bytes, 1 tape blocks
x srvr-2-0-sunos/waisgate/icons/binary.xbm, 533 bytes, 2 tape blocks
x srvr-2-0-sunos/waisgate/icons/eye.xbm, 272 bytes, 1 tape blocks
x srvr-2-0-sunos/waisgate/icons/eye2.xbm, 500 bytes, 1 tape blocks
x srvr-2-0-sunos/waisgate/icons/ftp.xbm, 524 bytes, 2 tape blocks
x srvr-2-0-sunos/waisgate/icons/image.xbm, 509 bytes, 1 tape blocks
x srvr-2-0-sunos/waisgate/icons/index.xbm, 530 bytes, 2 tape blocks
x srvr-2-0-sunos/waisgate/icons/menu.xbm, 527 bytes, 2 tape blocks
x srvr-2-0-sunos/waisgate/icons/movie.xbm, 530 bytes, 2 tape blocks
x srvr-2-0-sunos/waisgate/icons/sound.xbm, 530 bytes, 2 tape blocks
x srvr-2-0-sunos/waisgate/icons/telnet.xbm, 533 bytes, 2 tape blocks
x srvr-2-0-sunos/waisgate/icons/text.xbm, 527 bytes, 2 tape blocks
x srvr-2-0-sunos/waisgate/icons/unknown.xbm, 515 bytes, 2 tape blocks
x srvr-2-0-sunos/waisgate/sample/waisgate-announce.html, 3068 bytes, 6 tape blocks
x srvr-2-0-sunos/waisgate/sample/directory-of-servers.src, 1052 bytes, 3 tape blocks
x srvr-2-0-sunos/waisgate/sample/comp.infosystems.wais.src, 1298 bytes, 3 tape blocks
x srvr-2-0-sunos/waisgate/waisgate, 26503 bytes, 52 tape blocks
x srvr-2-0-sunos/waisgate/waisgate.conf, 1429 bytes, 3 tape blocks
x srvr-2-0-sunos/waisgate/waisexec, 548864 bytes, 1072 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/parseFile-main.c, 6831 bytes, 14 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/parseUtil.c, 6886 bytes, 14 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/parseUtil.h, 2803 bytes, 6 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/parseFile.c, 17702 bytes, 35 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/parseFile.h, 1424 bytes, 3 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/fieldInfo.c, 4550 bytes, 9 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/fieldInfo.h, 2556 bytes, 5 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/standardParsers.c, 36307 bytes, 71 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/standardParsers.h, 8591 bytes, 17 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/contribParsers.c, 40404 bytes, 79 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/contribParsers.h, 1169 bytes, 3 tape blocks

x srvr-2-0-sunos/parser-toolkit/src/localParsers.c, 17482 bytes, 35 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/localParsers.h, 1157 bytes, 3 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/ustubs.c, 4548 bytes, 9 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/ustubs.h, 1957 bytes, 4 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/list.c, 5874 bytes, 12 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/list.h, 2282 bytes, 5 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/cutil.c, 22638 bytes, 45 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/cutil.h, 6417 bytes, 13 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/futil.c, 27673 bytes, 55 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/futil.h, 4522 bytes, 9 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/date.c, 9013 bytes, 18 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/date.h, 1699 bytes, 4 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/panic.c, 824 bytes, 2 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/panic.h, 158 bytes, 1 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/waislog.c, 2405 bytes, 5 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/waislog.h, 2062 bytes, 5 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/cdialect.h, 2543 bytes, 5 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/version.h, 1039 bytes, 3 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/irdirent.h, 649 bytes, 2 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/Makefile, 2370 bytes, 5 tape blocks
x srvr-2-0-sunos/parser-toolkit/src/Parser-Manual.txt, 28020 bytes, 55 tape blocks
x srvr-2-0-sunos/parser-toolkit/Makefile, 1872 bytes, 4 tape blocks
x srvr-2-0-sunos/parser-toolkit/README, 514 bytes, 2 tape blocks
x srvr-2-0-sunos/parser-toolkit/lib/CVS/Repository, 42 bytes, 1 tape blocks
x srvr-2-0-sunos/parser-toolkit/lib/CVS/Entries, 772 bytes, 2 tape blocks
x srvr-2-0-sunos/parser-toolkit/lib/README, 174 bytes, 1 tape blocks
x srvr-2-0-sunos/parser-toolkit/lib/alphasort.c, 392 bytes, 1 tape blocks
x srvr-2-0-sunos/parser-toolkit/lib/freedir.c, 158 bytes, 1 tape blocks
x srvr-2-0-sunos/parser-toolkit/lib/ftw.c, 2515 bytes, 5 tape blocks
x srvr-2-0-sunos/parser-toolkit/lib/ftwtest.c, 555 bytes, 2 tape blocks
x srvr-2-0-sunos/parser-toolkit/lib/pdftw.h, 1025 bytes, 3 tape blocks
x srvr-2-0-sunos/parser-toolkit/lib/scandir.c, 2091 bytes, 5 tape blocks
x srvr-2-0-sunos/parser-toolkit/lib/scantest.c, 706 bytes, 2 tape blocks
x srvr-2-0-sunos/parser-toolkit/lib/stringtoany.c, 640 bytes, 2 tape blocks
x srvr-2-0-sunos/parser-toolkit/lib/trunc.c, 365 bytes, 1 tape blocks
x srvr-2-0-sunos/parser-toolkit/lib/Makefile, 1786 bytes, 4 tape blocks
x srvr-2-0-sunos/client-toolkit/src/sockets.c, 6774 bytes, 14 tape blocks
x srvr-2-0-sunos/client-toolkit/src/sockets.h, 708 bytes, 2 tape blocks
x srvr-2-0-sunos/client-toolkit/src/ustubs.c, 4548 bytes, 9 tape blocks
x srvr-2-0-sunos/client-toolkit/src/ustubs.h, 1957 bytes, 4 tape blocks
x srvr-2-0-sunos/client-toolkit/src/list.c, 5874 bytes, 12 tape blocks
x srvr-2-0-sunos/client-toolkit/src/list.h, 2282 bytes, 5 tape blocks
x srvr-2-0-sunos/client-toolkit/src/cutil.c, 22638 bytes, 45 tape blocks
x srvr-2-0-sunos/client-toolkit/src/cutil.h, 6417 bytes, 13 tape blocks
x srvr-2-0-sunos/client-toolkit/src/futil.c, 27673 bytes, 55 tape blocks
x srvr-2-0-sunos/client-toolkit/src/futil.h, 4522 bytes, 9 tape blocks
x srvr-2-0-sunos/client-toolkit/src/date.c, 9013 bytes, 18 tape blocks
x srvr-2-0-sunos/client-toolkit/src/date.h, 1699 bytes, 4 tape blocks
x srvr-2-0-sunos/client-toolkit/src/panic.c, 824 bytes, 2 tape blocks
x srvr-2-0-sunos/client-toolkit/src/panic.h, 158 bytes, 1 tape blocks
x srvr-2-0-sunos/client-toolkit/src/waislog.c, 2405 bytes, 5 tape blocks
x srvr-2-0-sunos/client-toolkit/src/waislog.h, 2062 bytes, 5 tape blocks
x srvr-2-0-sunos/client-toolkit/src/appConnCli.c, 7661 bytes, 15 tape blocks

x srvr-2-0-sunos/client-toolkit/src/appConnCli.h, 24248 bytes, 48 tape blocks
x srvr-2-0-sunos/client-toolkit/src/appConnMgr.c, 17380 bytes, 34 tape blocks
x srvr-2-0-sunos/client-toolkit/src/appConnMgr.h, 39224 bytes, 77 tape blocks
x srvr-2-0-sunos/client-toolkit/src/nConnMgr.c, 9615 bytes, 19 tape blocks
x srvr-2-0-sunos/client-toolkit/src/nConnMgr.h, 5405 bytes, 11 tape blocks
x srvr-2-0-sunos/client-toolkit/src/tcpConn.c, 6710 bytes, 14 tape blocks
x srvr-2-0-sunos/client-toolkit/src/tcpConn.h, 1479 bytes, 3 tape blocks
x srvr-2-0-sunos/client-toolkit/src/stdioConn.c, 5668 bytes, 12 tape blocks
x srvr-2-0-sunos/client-toolkit/src/stdioConn.h, 1485 bytes, 3 tape blocks
x srvr-2-0-sunos/client-toolkit/src/irfileio.c, 15232 bytes, 30 tape blocks
x srvr-2-0-sunos/client-toolkit/src/irfileio.h, 1383 bytes, 3 tape blocks
x srvr-2-0-sunos/client-toolkit/src/cdialect.h, 2543 bytes, 5 tape blocks
x srvr-2-0-sunos/client-toolkit/src/Makefile, 1604 bytes, 4 tape blocks
x srvr-2-0-sunos/client-toolkit/src/README, 2021 bytes, 4 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/connV2Cli.c, 24424 bytes, 48 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/connV2Cli.h, 2502 bytes, 5 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/recPrs.c, 26921 bytes, 53 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/recPrs.h, 845 bytes, 2 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/rpnBld.c, 5809 bytes, 12 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/rpnBld.h, 835 bytes, 2 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/cli.c, 26779 bytes, 53 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/cli.h, 2858 bytes, 6 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/connV2Mgr.c, 9491 bytes, 19 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/connV2Mgr.h, 2574 bytes, 6 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/berutil.c, 33592 bytes, 66 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/berutil.h, 7510 bytes, 15 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/tag.c, 5261 bytes, 11 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/tag.h, 10600 bytes, 21 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/diag.c, 9240 bytes, 19 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/diag.h, 940 bytes, 2 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/attr.h, 2166 bytes, 5 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/oid.h, 1438 bytes, 3 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/Makefile, 1135 bytes, 3 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z3950V2/README, 268 bytes, 1 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/conn1988Cli.c, 13673 bytes, 27 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/conn1988Cli.h, 3004 bytes, 6 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/docid.c, 3275 bytes, 7 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/docid.h, 909 bytes, 2 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/docid-io.c, 6241 bytes, 13 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/docid-io.h, 300 bytes, 1 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/wprot.c, 61377 bytes, 120 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/wprot.h, 9502 bytes, 19 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/wutil.c, 41212 bytes, 81 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/wutil.h, 1077 bytes, 3 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/zprot.c, 29592 bytes, 58 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/zprot.h, 5772 bytes, 12 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/zutil.c, 14185 bytes, 28 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/zutil.h, 8045 bytes, 16 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/ztype1.c, 7290 bytes, 15 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/ztype1.h, 2961 bytes, 6 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/transprt.c, 6573 bytes, 13 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/transprt.h, 390 bytes, 1 tape blocks

```
x srvr-2-0-sunos/client-toolkit/src/z39501988/wmessage.c, 2428 bytes, 5 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/wmessage.h, 998 bytes, 2 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/conn1988Mgr.c, 8042 bytes, 16 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/conn1988Mgr.h, 1537 bytes, 4 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/Makefile, 1170 bytes, 3 tape blocks
x srvr-2-0-sunos/client-toolkit/src/z39501988/README, 246 bytes, 1 tape blocks
x srvr-2-0-sunos/client-toolkit/demo/demo-main.c, 26351 bytes, 52 tape blocks
x srvr-2-0-sunos/client-toolkit/demo/Makefile, 1107 bytes, 3 tape blocks
x srvr-2-0-sunos/client-toolkit/demo/README, 640 bytes, 2 tape blocks
x srvr-2-0-sunos/client-toolkit/lib/README, 357 bytes, 1 tape blocks
x srvr-2-0-sunos/client-toolkit/include/README, 561 bytes, 2 tape blocks
x srvr-2-0-sunos/client-toolkit/include/cdialect.h, 2543 bytes, 5 tape blocks
x srvr-2-0-sunos/client-toolkit/include/cutil.h, 6417 bytes, 13 tape blocks
x srvr-2-0-sunos/client-toolkit/include/list.h, 2282 bytes, 5 tape blocks
x srvr-2-0-sunos/client-toolkit/include/date.h, 1699 bytes, 4 tape blocks
x srvr-2-0-sunos/client-toolkit/include/waislog.h, 2062 bytes, 5 tape blocks
x srvr-2-0-sunos/client-toolkit/include/appConnMgr.h, 39224 bytes, 77 tape blocks
x srvr-2-0-sunos/client-toolkit/include/appConnCli.h, 24248 bytes, 48 tape blocks
x srvr-2-0-sunos/client-toolkit/include/nConnMgr.h, 5405 bytes, 11 tape blocks
x srvr-2-0-sunos/client-toolkit/INSTALL, 440 bytes, 1 tape blocks
x srvr-2-0-sunos/client-toolkit/Makefile, 1648 bytes, 4 tape blocks
x srvr-2-0-sunos/client-toolkit/README, 1068 bytes, 3 tape blocks
```

Also note that in this example, the current WAIS software release is wais-2-0. The actual release which you are unpacking may have a different name.

# Preparing for Upgrades and Releases

Before we examine the release itself, let's take a look at how WAIS releases are organized. This is just a suggested organization, if your site has different requirements, feel free to modify this structure as you see fit.

As it is shipped, each WAIS release fits completely into one directory structure. The name and version number of the release is encoded in the name of the directory. In our example, the name is wais-*majorversion-minorversion*. As future releases are made, the version number is incremented. This way, you can have several versions of WAIS available on your system at the same time. Furthermore, you don't need to worry about overwriting old releases when you upgrade.

When you set up your WAIS server and WAIS databases, you won't want to encode the version number in each configuration file, since that would mean reconfiguring each file when a new release is installed. Instead we recommend making a symbolic link which maps the name of

the release directory into a standard name used by the configuration files. In our example, we'll use current as the standard name.

Make sure you are in the top level WAIS directory, and type:

**% cd /wais**

and make a link

**% ln -s wais-***majorversion-minorversion* **current**

When a new release is installed (e.g. wais-2-0), you can switch over to the new software by removing the old link, and making a new one.

**% rm current**
**% ln -s wais-***newmajorversion-newminorversion* **current**

This procedure makes sure you have a symbolic link from the /wais/current directory to the most recent WAISserver software release.

Throughout the rest of this manual, we will use the /wais/current directory name when giving file pathnames included in the WAISserver software release. If you choose a different directory for your WAISserver software, substitute the appropriate directory name for your site.

# Installing the WAIS Programs & Online Manuals

To make the WAIS programs available to you on the command line without having to type the full pathname of each program, you will need modify your **PATH** environment variable.[1] To make the online reference manuals accessible through the UNIX **man** command, you will also need to change the **MANPATH** environment variable. It is advisable to do this after making the link described in the previous section, so that you don't have to do it again every time new software is installed. To modify these environment variables, you will need to edit your .cshrc file, which is found in your home directory. Add the following line to the bottom of the file:

[1]The installation described applies to C shell (csh) users only. If you use another shell (eg. borne shell, korn shell, etc), you will need to modify installation to suit your shell.

```
setenv PATH /wais/current/bin:$PATH
setenv MANPATH /wais/current/man:$MANPATH
```

Make sure the .cshrc file is saved, then use the following command to load the new version into your UNIX shell:

```
% source ~/.cshrc
```

You can then try using one of the WAIS programs by typing:

```
% waisserver
USAGE: waisserver
        [-port] (the tcp-ip port, defaults to 210)
        [-directory] (use directory as the source of databases)
        [-e logFile] (the file to write log info to)
        [-u user] (if started as root, setuid to user after startup)
        [-v] (print the version of the software)
```

You can then type,

```
% man waisserver
```

to view the manual page on **waisserver**. For your convenience, the UNIX manual pages are reprinted in Chapter 9, entitled "Command Reference."

If you want other users to be able to use the WAIS programs and manuals too, ask your system administrator to include the **setenv** command in your site's shared .cshrc file.

# 3

# The WAIS Directory Structure

This chapter is a guided tour of the WAIS release structure. It will give you a feel for where the software components can be found, and it will serve as an introduction to administering the WAIS system as a whole. The directory structures are simply our recommendations. Feel free to adjust them to meet your own needs. If you are not sure what your needs are, don't worry, it's easy to set up a system in the default configuration, then change it around once you understand your unique constraints.

There are five main components in a WAIS installation. Each WAIS component is located in a separate directory under the root directory. In our example installation, we have selected **/wais** as our root directory.

**/wais/current**

This directory contains the most recent release of the WAIS programs, the WAIS documentation, the UNIX-formatted manual pages, some sample databases to play with, and utilities. The contents of this directory is supplied for you with each new release.

**/wais/data**

This directory consists of your original data which is supplied and maintained by you. If your WAIS server will be handling multiple databases, this directory is typically organized as a set of subdirectories, one for each data set.

**/wais/indexes**

This directory contains WAIS indexes. The indexes help the WAIS server to quickly search and retrieve documents from your databases. For each database in the **data** directory, a corresponding index directory is created and maintained by you using the waisparse and waisindex programs. Thus this directory mirrors the subdirectory structure of the **data** directory.

**/wais/logs**

Your WAIS server records all client transactions in the log files of this directory. This information can be used for billing and statistical analysis.

**/wais/remote-dbs**

This directory is made up of the source definitions of public WAIS databases. It is used by the get-remote-dbs script in the /wais/current/utilities directory. The script maintains a local copy of the public directory of servers. This is only applicable if you periodically run the script and have a connection to the Internet.

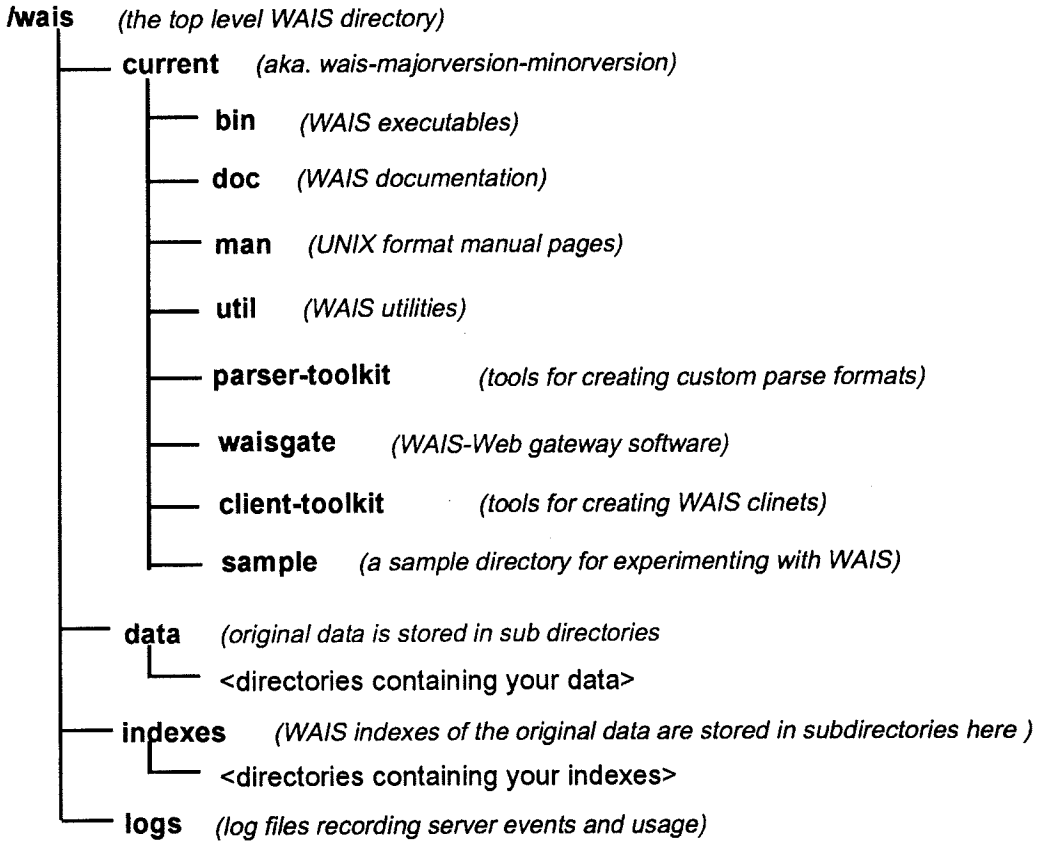



*Figure 2: The WAIS Directory Structure*

This chapter describes these directories in detail. While working through this chapter, refer to Figure 2 for a close-up view of the overall structure.

# The current Directory

The **/wais/current** directory has the WAIS software installation. It contains four subdirectories:

**bin**

This directory contains the WAIS programs: **waisparse**, **waisindex**, **waisserver**,**waislookup**,**waisreporter**, and waisdelete.

**doc**

This directory contains the *WAISserver Administration Manual.*

**man**

This directory contains the UNIX-formatted command reference pages (known as man pages) for each of the WAIS programs located in the **bin** directory.

**util**

The utility directory contains programs and data you may find useful.

# The data Directory

The **/wais/data** directory is the home of your original data collection. You can store these files anywhere, but if they are moved or modified, the index will need to be rebuilt.[2] Files that are already part of your system need not be moved, but if you are collecting a new batch of files for indexing, it's convenient to have a standard place to put them.[3] Our convention is to collect the data files in the data subdirectory of the main WAIS root directory (e.g. /wais/data). If this is the first time you are installing WAIS, this directory will not exist yet. You can create it with the following commands.

---

[2]Even after indexing, the original data files are critical to the functioning of a database. These are the files from which data is retrieved when a user requests a document. Therefore they must not be moved or modified without an accompanying index rebuild. Furthermore, their names must stay the same so that the server find them.

[3]The contents of /wais/data could be large, and may not fit on one disk. To solve this, you may put the data on another disk, and make a symbolic link to the files.

```
% cd /wais
% mkdir data
```

Within the data directory, build a subdirectory for each set of data files you will serve. For example, if you have a set of files containing information on customer contacts, make a special directory for it. It can be called contacts (/wais/data/contacts). You could also have a directory for your staff mailing list (/wais/data/staff), and another for your company's proposals (/wais/data/proposals).

# The indexes Directory

Just as the data is usually stored in it's own directory, the indexes are also stored in their own directory. Besides convenience, this organization makes serving the data easier (for more information, see the Chapter 5, Setting Up a WAIS Server). In our examples we will call the directory /wais/indexes. If this is the first time you are installing WAIS, the indexes directory will not exist yet. You can create it with the following commands.

```
% cd /wais
% mkdir indexes
```

When you run the waisindex program as described in the next chapter, it will automatically create one subdirectory within the **/wais/indexes** directory for each database that you have stored in a /wais/data directory. The /indexes subdirectories will have the same names as the /data subdirectories. For example, if you have the contacts, staff, and proposals databases described above, waisindex will create directories named /wais/indexes/contacts, /wais/indexes/staff, and /wais/indexes/proposals. When you follow the instructions in Chapter 4, "Building a WAIS Database," the index directories will be created and given contents.

# The logs Directory

The server logs should all be kept in a single directory. We suggest **/wais/logs**. If this is the first time you are installing WAIS, the log directory will not exist yet. You can create it with the following commands.

```
% cd /wais
% mkdir logs
```

The server records entries in the log file located in this directory. Chapter 7, Monitoring WAIS Usage, will introduce automated scripts which keep a separate log file for each day, and send a daily usage report to the system administrator.

# 4

# Building a WAIS Database

This chapter describes how to create a WAIS database, starting with a collection of data. WAIS is an extremely flexible system that accomodates data in any format. Therefore, different command options are used depending on the nature of the database you are building. It is important to decide which options are best suited to your particular database.

Before using the commands explained here, read through this chapter and decide which of the several kinds of databases described you will be building. Also, look through the section entitled "Indexing Customizations," in Chapter 6, "Customizing a WAIS System." You may also want to incorporate some of advanced features described there.

## What is a WAIS Database?

A WAIS database is made up of two main components: the original set of data, and a WAIS index of this data. The original data set is supplied by you. The WAIS index is a series of files generated by the WAIS programs. The WAIS index files facilitate fast search and retrieval of the data. Taken together, the data set and the WAIS index are the essential ingredients making up a complete WAIS database system.

Typically, the original data is made up of a set of documents, headlines, and words. A document is the smallest retrievable element of the database. For example, a database may contain a volume of journals, where each article of the journal is a separate document. Another example is electronic mail, where each mail message is a different document. A document may also be made up of text, images, or video, or a combination thereof. In a database of resumes, for instance, each document may contain a resume of a person in your company's technical marketing division, a picture of that person, and possibly even a short video of a marketing presentation done by that person.

Each document has a headline. A headline is one or more words that embody the main idea behind the document. When a client sends a question to a server, the server responds with a list of headlines that represent the most relevant documents related to the client's question. Each headline is automatically extracted for you from the data.

In addition to a headline, each document is also associated with one or more words. These words constitute the text of the document. Together, the headline and the words are used to determine how relevant a document is to a client's question.

A WAIS index is constructed by using the waisparse and waisindex programs to create a WAIS Index. The parser takes the original data and separates it into documents, headlines, and words. The indexer takes the information generated by the parser and creates the WAIS Index. The waislookup program is used to check the search and retrieval of information in the database.

# What's in a WAIS Index?

A WAIS index is made up of several index files created by the waisindex program and several auxiliary files created by the WAIS administrator. They are all stored together in a common directory. This section describes each of these files and explains how they are used for the search and retrieval of documents in a database.

## Index Files

Unless otherwise noted, index files are binary files and cannot be edited by hand.

**Dictionary (.dct)**

The Dictionary File contains a list of all the words used in a database.

**Inverted File ( .inv)**

The Inverted File contains a list of all the words in the database, and for each word, it lists all the documents which contain that word.

**Document Table ( .doc )**

The Document Table contains a record of each document in the database.

**Headline Table ( .hl)**

The Headline Table contains headlines of all the documents in the database.

**Filename Table ( .fn)**

The Filename Table contains a list of the filenames of the original data files.

**Catalog (.cat )**

The Catalog File contains a human readable list of headlines and document identifiers for some or all of the documents in the database. This list may be returned to a user whose search has gone poorly, as an aid to help them understand the contents of the database.

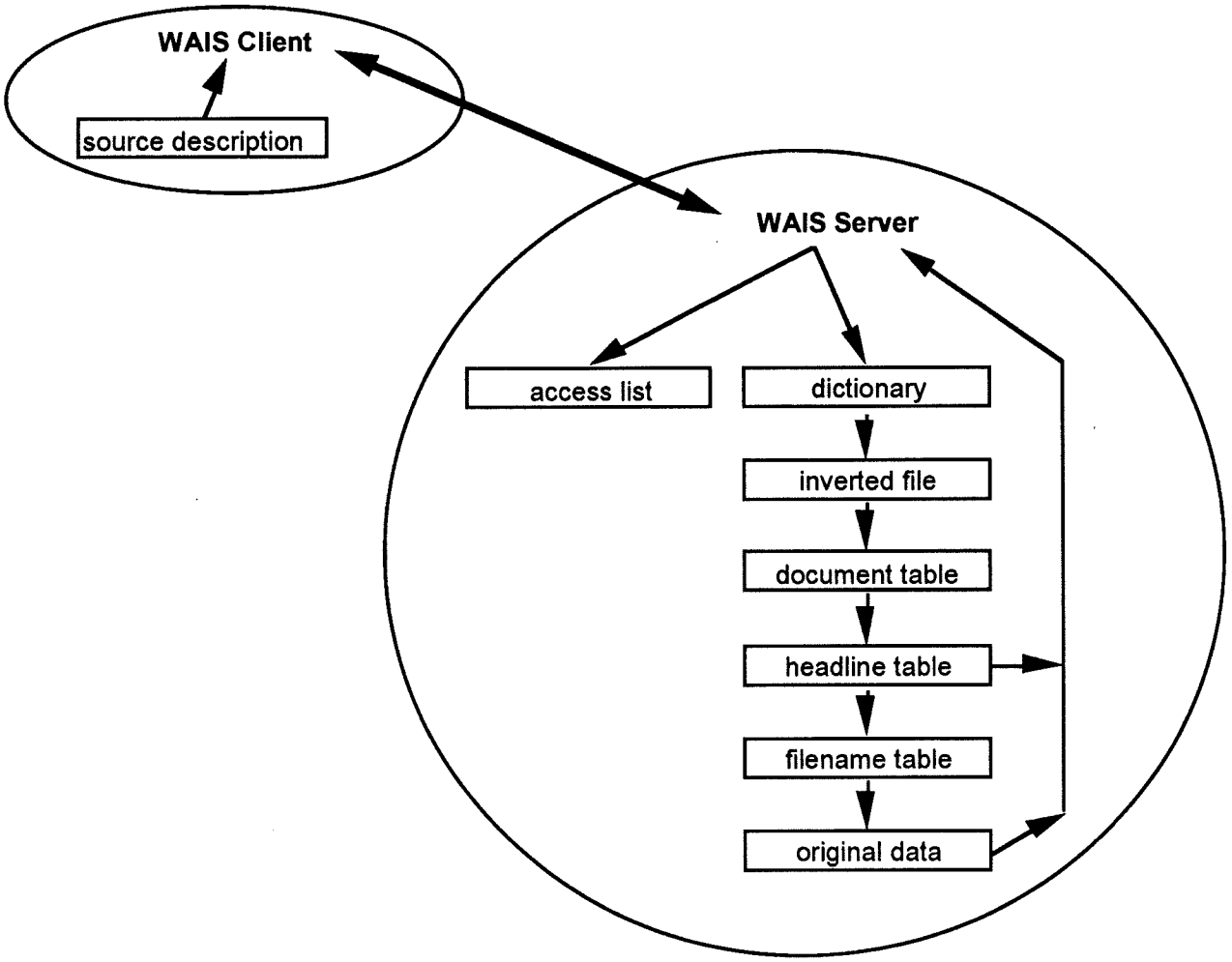



*Figure 3: How index files are used during a search.*

The interaction between the WAIS index files is illustrated in Figure 3. A client process uses information from the source description file to find and contact the server. The server checks the access list to make sure the client has permission to access this database. If so, the server process takes the words from the client's query and looks them up in the database's dictionary file. The dictionary file provides pointers into the inverted file, where, for each word, there is a list of pointers into the document table corresponding to the documents that contain that word. The information from the inverted file is used to look in the document table, which gives a pointer to the headline table, which in turn gives a pointer to the filename table. Finally, the information from the filename table is used to access the original data. A list of headlines and relevance scores is returned to the client process for display to the end user.

## Auxiliary Files

The index directory can also include auxiliary files, which are optionally created or modified by the WAIS Administrator. Unlike the index files proper, none of these auxiliary files are overwritten if the database index is updated or rebuilt. They are all ascii files that may be edited with any text editor.

**Source Description ( .src)**

The Source Description File describes the database and the server. It is used by the client process to contact the server and search the database. Source description files are distributed to clients by the directory-of-servers. They are ASCII files created in template form by waisindex and must be edited by hand. The administrator corrects the automatically generated information as necessary and adds a textual description of the data set. The source description file is the default file returned when an end user types 'help' as a query.

**Access List (.acc )**

The Access List contains the addresses of all machines that are allowed to search the database. It is an editable file that you create as described in detail in Chapter 6, "Customizing a WAIS System."

**Notices File (index.not)**

This file contains special notices from the WAIS database administrator that are displayed at the beginning of every results list.

**Help File ( index.hlp )**

When a WAIS client user types **help** as the query, or issues an empty query, if the file **index.hlp** is present in the WAIS index directory, its contents — instead of the .src file contents — will be returned to the client for display to the user.

**Configuration File (index.wc )**

The database-specific WAIS configuration file, **index.wc**, defines various run-time parameters such as the headline for query reports, the email address of the administrator, and whether or not the multi-database mechanism is in use.

It is also possible to define a global configuration file that is shared by multiple databases. See the section entitled "Configuration Files," in Chapter 6, "Customizing a WAIS System."

# Preparing to Make an Index

Before creating a WAIS index, there are several decisions that you will need to make.

- How much disk space your original data requires, and how much additional space you will need to perform indexing and to store your WAIS index.
- The best location for your WAIS index. This decision consists of selecting the directory for your index, and the machine on which the index will be stored.
- Where the documents, headlines, and words exist in the original data format. This will determine the selection of the parse format for the files in your database.
- The format your data should be displayed in to a client. Examples of display formats include ASCII text, Microsoft Word and GIF Images.

Each of these issues is addressed in detail in the following sections.

# Disk Space Requirements

To create a WAIS index for your data, you will need to ensure that you have enough free disk space to build the indexes and to store the resulting index files. This section provides guidelines for roughly calculating your storage requirements. The calculation is broken into three parts: storage of the original data, storage during the process of building the index files, and storage of the index files.

First, you will need to measure how many megabytes of storage your original data requires. To do this, use the **du -s** command on your database directory to summarize how much disk space your database currently requires.

```
% du -s /wais/data/mydata
534073
```

In this example, approximately 534 megabytes of disk space is required to store the data in the directory called /wais/data/mydata. To generalize the calculation, let us assume that the amount of disk space required is *n* megabytes.

Next, you will need to determine how much space is required during the creation of the index files. In general, building an index for *n* megabytes of data requires an additional *n* megabytes of disk space. Half of this space is necessary for temporary storage during the building process, and the other half is required to store the finished index files.

After the index files have been built, the amount of disk space required to store the index files is 1/3 to 1/2 of *n*. Thus, the total amount of disk space required to store a full WAIS database is usually no more than 1.5*n*. This corresponds to the sum of the space required to store the original data, *n*, plus the space required to store the index files, which is typiclly less than 0.5*n*.

# Index Location

When deciding on the location of the index files, there are two issues to consider. The first issue concerns the directory in which the indexes are stored, and the second issue concerns the machine on which these indexes physically reside. Both issues are discussed in detail below, with our recommendations on how you can best address these issues at your site.

For public or shared databases, we recommend storing the indexes in one common directory. As described in Chapter 3, The WAIS Directory Structure, the suggested location of this directory is /wais/indexes. If your indexes are stored on different disks, you can use symbolic links make it appear that the indexes are located in the same directory.

The reason for placing the indexes in a common directory is so that clients requesting information from a database do not have to know the full pathname of the index for that database. When the server is set up as described in Chapter 5, "Setting Up a WAIS Server," this common directory is specified with the **-d** switch. When a client requests information from a database on the server, only the name of the database is required and not the full pathname. This allows you to

change the location of the index directory without having to notify the clients. Instead, you only need to update the server's **-d** switch.

Within the /wais/indexes directory, there should be one directory for each database that you index. Index files may be moved around, as long as all the files for a given database are in a single directory, and a link to the directory is maintained in /wais/indexes. This means that indexes can be created on disks other than the one containing the WAIS software. This is very important when building large databases.

It is highly recommended that indexes be physically located on the disks of the server machine. This strategy is recommended primarily for performance reasons. When a client sends a request to a server machine, the server searches through the index files to determine which documents are most relevant to the information request. This process is most efficient if the index files are local to the server.

# Parse Format

The parse format tells the WAIS parser how each file in your database is organized. More specifically, the parse format specifies how the parser should distinguish each document in your database. For each document, it also specifies how to determine the headline, content-bearing words, and fields. For example, the parse format determines if a file contains a single document or a set of documents. It also determines if the headline is the filename or a string embedded in the document.

A parse format which best matches your database should be selected from the formats described in this section. In most cases, the format required by your database will be a supported format. In other cases, you may need to reformat your data to match an existing format, or build your own parse format.

**dash**

The **dash** parse format is useful if a single file contains multiple distinct documents. In the dash format, each document is separated by a line containing a minimum of 20 dash characters, "-". The line following the dashed line is expected to contain a headline, followed by the text of the document.

**dvi**

The **dvi** parse format is for Device Independent Printer Output files. The filename is used for the headline, and the contents of the file supply the words of the document.

**filename**

The **filename** parse format treats each file as a single document, and uses the filename as the headline. The contents of the file however are not parsed unless the **-contents** switch is present. This format is useful for databases constructed of many individual binary files, for example, whose contents are not words.

**first-line**

The **first-line** parse format specifies that each file contains a single document, and that the first non-blank line of the file is the headline, and the remainder of the file is parsed as words.

**first-words**

The **first-words** parse format is similar to first-line, except that the headline is the first 100 non-whitespace characters in the file.

**gif**

The **gif** parse format is for Graphics Interchange Format files. The file is considered to be a single document where the filename is used as the headline, and there are no words in the document. Image files, such as gif, can be associated with a text file and considered as a single document by the WAIS parser, indexer, and server. (See the **-assoc** option to **waisparse**).

**html**

The html parse fomat is for files using the Hyper-Text Markup Language. This is the standard format for files used by World-Wide Web servers and clients.

**mail-digest**

The **mail-digest** parse format is for standard Internet mail digest files. A mail-digest file contains one or more electronic mail messages, in which each mail message is parsed as a separate document. The subject line of the mail message is the headline, and the body of the message contains the words of the document. In addition, from, to, cc, subject, sender, and date are recognized as fields.

**mail-or-rmail**

The **mail-or-rmail** parse format is used for UNIX mail files that conform to RFC822. A mail file is a single file containing one or more electronic mail messages, where each mail message is parsed as a separate document. The subject line of the mail message is the headline, and the body of the message contains the words of the document. In addition, from, to, cc, subject, sender, and date are recognized as field names and their contents as field information.

**netnews**

The **netnews** parse format is used for Internet Network News, where each Network News or Read News file contains one or more news messages. Each news message is parsed as a document, where the subject line is the headline, and the body of the message contains the words of the document. In addition, subject, newsgroup, from, and date are recognized as field names and their contents as field values.

**one-line**

The **one-line** parse format is a simple format that treats each line of a file as a separate document. The line also forms the headline for that document.

**paragraph**

Like the dash format, the **paragraph** parse format is useful if a single file contains multiple distinct documents. In the paragraph format, each paragraph is separated by one or more blank lines. The first line of each document is the headline, which is followed by the text of the document.

**pict**

The **pict** parse format is for Apple PICT image file formats. The file is considered to be a single document where the filename is used as the headline, and there are no words in the document. Image files, such as pict, can be associated with a text file and considered as a single document by the WAIS parser, indexer, and server. (See the **-assoc** option to **waisparse**).

**ps**

The **ps** parse format is for PostScript files. The filename is used for the headline, and the contents of the file supply the words of the document.

**source**

The **source** file format (.src) is a file format generated by the WAIS indexer and used for the directory of servers. The file typically contains information about the database, and is parsed exactly like the text file format.

**text**

In **text** format, each file is treated as a single document, the filename is used as the headline, and the contents of the file is parsed as words. This format is useful for databases constructed of many individual files. This is the default parse format.

**tiff**

The **tiff** parse format is for tagged interchange file formats. The file is considered to be a single document where the filename is used as the headline, and there are no words in the document. Image files, such tiff, can be associated with a text file and considered as a single document by the WAIS parser, indexer, and server. (See the **-assoc** option to **waisparse**).

The display format determines how a client should display a retrieved document. In many cases, the document is a simple text file, and thus has no special display needs. In other cases, the document may be in a specific format that should be displayed with a special display program. For example, suppose a document was generated using Microsoft Word. The client that retrieves this document must be told that the document is in Microsoft Word format in order to display it correctly. To do this, a display format of MS-WORD is given to the parser by using the -display option to waisparse.

The parser associates a display format with each document. The parser passes the display format through to the indexer, which in turn stores it for later use by the server. When a client initiates a query, the server sends the client both a list of headlines and a list of display formats in which the documents are available. It's up to the client to decide which, if any, of the display formats it can display.

Examples of display formats supported on many existing WAIS clients are described in Table 1.

| Display Format | Description of Format |
|---|---|
| DVI | Device-Independent Printer Output |
| GIF | Graphics Interchange Format<br>CompuServe Images |
| HTML | Hyper-Text Markup Language |
| MIME | AT&T Multimedia Document |
| MS-EXCEL | Microsoft Excel Spreadsheet |
| MS-POWERPOINT | Microsoft PowerPoint Slides |
| MS-WORD | Microsoft Word Document |
| PERSUASION | Aldus Persuasion |
| PICT | Apple PICT Image |
| PS | PostScript |
| QUICKTIME | Apple Quicktime Movie |
| TEXT | ASCII Text |
| TEXT-FTP | Special FTP File Format |
| TIFF | Tagged Interchange File Format (.tif)<br>A Universal Raster Image Format |
| WQST | WAIS Question Format (.qst) |
| WSRC | WAIS Source Format (.src) |

*Table 1: Display Formats*

The exact display format you select depends on the display format supported by the WAIS clients that will be accessing your database. For this reason, you may want to check with the client program to determine what display formats it supports.

If a display format is not specified, the default value of the display format is dependent on what is specified as the default for the parse format. If the parse format is **gif**, for example, then the display format defaults to **GIF**. Otherwise, the default display format is TEXT.

# Building an Index

A WAIS index is created from your original data using the **waisparse** and **waisindex** programs. See the UNIX manual pages for these commands in the last chapter of this manual, entitled "Command Reference," or type

**% man waisparse**

and then

**% man waisindex**

for a complete description of all the options available with these commands.

The **waisparse** program takes a collection of documents in the form of files and directories or as a stream of data. It separates the data into distinct documents and outputs them as a stream. A stream is, as a sounds, a continuous flow of data.

For each document, the parser identifies a headline, the most recent modification data, and the content-bearing words. Optionally, fields and document keys may also be identified.

The **waisindex** program takes the stream of documents output by **waisparse** and creates an index for fast search and retrieval of these documents. Properly executing these two programs will complete the building of your WAIS database.

To build a WAIS index, use the **waisparse** command piped into the **waisindex** command. The command line is:

**waisparse -parse** *parse-format* | **waisindex -d** *database-indexdir*

where *parse-format* is the parse format for your data, *filespec* identifies the data files, and *database-indexdir* is the pathname of the index files. If the directory specified by *database-indexdir* does not exist, the indexer creates it. If it already exists, the indexer overwrites the index files — but leaves the auxillary files unchanged.

## Single-Document Files

In the simplest case, a database may consist of one document per file, where the headline of each document is the same as the filename. The command line

```
% waisparse /wais/data/resume/*.txt |
      waisindex -d /wais/indexes/resume
```

is an example of building an index from data stored in single-document files. Here, the waisparse command takes only a *filespec* argument: /wais/data/resume/*.txt, which specifies all files in the /wais/data/resume directory that have the **.txt** filename extension. The | character is a *pipe*; it instructs UNIX to send the output of waisparse directly to waisindex. The waisindex program then builds a index in the specified directory, /wais/indexes/resume.

Notice that no -parse or -display options are used. Instead we let waisparse use the default parse and display formats: **text** and **TEXT**, respectively.

Now, if you wanted to index all the files in the resume directory as well as all the files in all its subdirectories, you would do the following:

```
% waisparse -r /wais/data/resume |
      waisindex -d /wais/indexes/resume
```

The optional **-r** switch tells the parser to recursively descend the /wais/data/resume directory, visiting each subdirectory in turn.

As another example, consider indexing a set of C source code files. In this case, programmers can use the WAIS system to quickly search through large amounts of source code spread across multiple development directories. As above, each source code file is considered a single document.

The command line

```
% find /dev/src -name '*.[ch]' -print | waisparse - |
      waisindex -d /wais/indexes/source-code
```

first finds all the files under **/dev/src** and the subdirectories of **/dev/src** that end in **.c** or **.h**. It then pipes the names of those files to **waisparse**. The - option tells **waisparse** to read the list of files to be parsed from the standard input. (Additional data files may be specified after the - character.) Finally, the stream of documents produced by **waisparse** is piped into **waisindex**, which creates a set of index files and places them in the directory specified by the -d option **/wais/indexes/source-code** .

## A Multi-Document File

Consider the case in which several documents are contained in a single file. Only the single filename needs to be specified in the call to **waisparse**. For example, the following command will index a personal electronic mail file.

```
% waisparse -parse mail-or-rmail /usr/smith/RMAIL |
      waisindex -d /usr/smith/indexes
```

Notice that Smith did not use the **/wais/indexes** directory as the argument to the **waisindex -d** switch. This is important if you wish to maintain your own personal WAIS database.

## Multiple Parse Formats

If your database requires multiple parse formats, you can specify each format in the command line. For example, consider a collection of

resumes inwhich a subset of the resumes use the filename as the headline and another subset are in Microsoft Word format. The command line might look like this:

```
% waisparse -parse first-line
        /wais/data/resume/*.txt
        -parse text -display PS
        /wais/data/resume/*.doc |
        waisindex -d wais/indexes/resume
```

The first set of resumes have a parse format of **first-line** and a default display format of **TEXT**, and the second set of resumes have a parse format of **text** and a display format of **PS**. Each time a new **-parse** switch is encountered, the display format is reset to the default value for that parse format. Thus, in the event that your display format is different from the default value, it is important to list the parse format before specifying the display format.

## Multi-Display Documents

A single document may be made up of multiple files, where each file has a different display format. One of the files is designated as the primary file. It contains the headline and the words that are parsed and indexed. The other files of the document are called the supporting files, and are specified with the **-assoc** switch to **waisparse**.

For instance, you may want to associate an image file with a text file describing it. When a client retrieves the document, the client may give the user the choice of viewing the image format, the text format, or both. The command line for this might look like:

```
% waisparse -parse first-line
        -assoc tif TIFF
        /wais/data/resume/*.txt |
        waisindex -d /wais/indexes/resume
```

Here the **-assoc** switch is used to associate a TIFF image, stored in files with a **.tif** extension, with a text file, where the text files are designated as the primary files. For example, the primary file **/wais/data/resume/joe.txt** is matched with the image file **/wais/data/resume/joe.tif**.

Multiple display formats are equally useful in cases where documents exist in multiple formats. A technical report, for example, may be available in text, Postscript, and Microsoft Word format and you may want to make all three formats available to the user. This can be done with a command such as:

```
% waisparse -assoc ps PS -assoc doc MS-WORD
        /wais/data/tech-reps/*.txt |
        waisindex -d /wais/indexes/resume
```

Here, the first **-assoc** switch indicates that Postscript files use the file extension **.ps** and the PS display format. The second **-assoc** switch identifies Microsoft Word files as those with **.doc** extensions and using the MS-WORD display format. The files that are parsed and indexed are specified as **/wais/data/tech-reps/*.txt** using the default **text** parse format. The Postscript and Microsoft Word format files, **/wais/data/tech-reps/*.ps** and **/wais/data/tech-reps/*.doc**, respectively, are associated with these text files.

## The .src File and the Directory of Servers

A directory of servers is a WAIS database made up of **.src** files. Each *database*.src file contains information about a particular database. Once you have built the index for your database, there will be an *database*.src file in your index directory. Using your favorite editor, open it and take a look. A sample *database*.src file is shown below.

```
(:source
  :version 3
  :ip-address "129.362.28.5"
  :ip-name "server-name:whereever.com"
  :tcp-port 210
  :database-name "/wais/indexes/mydata"
  :maintainer "whoever@whereever.com"
  :description
"This is where you put a textual description of the
database you are defining. Be sure to include a description of any tagged fields you may
have defined for the database. This is the information users will turn to to decide
whether or not to search your database."
)
```

The values in each field will be different from those shown above. Check them and correct any incorrect values. The waisindex program does its best to fill in the correct information, but you probably need to change something. In particular, be sure to check the ip-address, ip-name, and tcp-port values. Also check the email address for the official maintainer. Next, add a textual description of the database. It should be placed after the information about when and by whom the database was created and within the final quotation marks. The text you put here is displayed to end users who are consulting a directory of servers looking for databases to search. If your database contains fields, list and describe them. (For more about fielded databases, see the section "Tagging Database Fields," in Chapter 6, "Customizing a WAIS System.")

You can create your own directory-of-servers database by parsing and indexing the *database*.**src** files in your index directory. For instance in the command line:

```
% waisparse -parse source /wais/indexes/*/*.src |
        waisindex -d /wais/indexes/directory-of-servers
```

tells **waisparse** to use the source parse format to parse all the files **/wais/indexes/*/*.src**. Then, waisindex creates an index to allow all the source information to be searched and places it in **/wais/indexes/directory-of-servers**. Now, when a user specifies the directory or servers as the database to search, her query will be run against this database of databases.

There is a public directory of servers database that lists all publicly available WAIS databases. Both commercial and non-profit databases are listed. Client processes extract information from the directory of servers in order to find and connect with WAIS servers. To request that your database be added to the master list, send an email message to directory-of-servers@wais.com. Your message should contain an exact copy of your *database*.src file. Please send one message for each *database*.src file you would like included in the directory. The public directory of servers may be retrieved periodically by running the /wais/current/utilities/get-remote-dbs script.

# Database Testing and Troubleshooting

The **waislookup** program can be used as an interactive interpreter for performing search, retrieval, and relevance feedback directly on a database without using the server. When bypassing the server in this way it is used to test and troubleshoot WAIS indexes. To test your newly built WAIS index, type

**waislookup -d** *index-path*

at the UNIX prompt, where *index-path* is the same directory name that you specified in the **-d** switch to **waisindex**. If a search fails, check for errors from **waisparse** or **waisindex**.

# Database Maintenance

## Using Cron to Do Your Work

Depending on the nature of your database, a number of the tasks described in this section may need to be performed on a periodic basis — perhaps daily. Almost any command you can issue at the command line can be wrapped into a shell script and executed on a regular basis by the UNIX cron facility. We recommend this approach if you are maintaining a database that requires predictable, repetative updates on a regular basis.

## Creating Additional Databases

To create additional WAIS databases, simply follow the same procedure you did to create the first database. The server does not require any modifications. Remember, however, you will need to reindex your local directory-of-servers database and — if you want to make a WAIS database publicly accessible, notify the public directory of servers.

## Using -nice when Indexing a Data Set

The **waisindex** command has a **-nice** option that keeps it from hogging the computer on which you run it. This can be especially useful if, for example, you need to index a large set of data during a time when users need quick response time from the server you are using. The syntax for the -nice option is:

% **waisindex -nice** *how-nice* **-d** *database-index-path*

where *how-nice* is a number between -20 and 20. The higher the number, the nicer the process — in the sense of reliquishing more time to other processes. Only a UNIX superuser (someone with permission to log in as root) can successfully specify a negative number (effectively *not* being nice).

## Moving a Database

Moving a database involves two main issues: moving the original data, and moving the WAIS index. If you move the original data, reindexing is required — unless you create a symbolic link from the old location to the new location. If a data set is moved without reindexing and without being replaced by a symbolic link to the new data location, then

subsequent queries will always result in one document with the headline *document not available* and length 0, but the correct score. The log file will record "Search returned document in unreadable file *filename*". For more information about log files, see Chapter 7.

Moving a WAIS index does not require reindexing. All that is required is renaming the directory and modifying the dbname field in the .src file. If the database is part of a directory of servers database, you will need to reindex the directory of servers database. In any event, you will also need to check that the permissions for accessing the new location are still valid for the original data and for the WAIS index.

## Incremental Indexing

If you need to add, delete, or modify the documents in a WAIS database, there are a variety of WAIS commands that allow you to update your database index without reindexing all the data and without requiring that users stop using the datebase.

The **waisindex** command has an **-append** option, which is used when new files are added to a data set or when old files are modified. Also, there is a **waisdelete** command that goes into the WAIS index and selectively "turns off" all references to designated data documents.

For those who are administrating a very volatile set of information files, advanced options to waisindex offer more fine-grained control over updates. These include the **-merge**, **-replacewith**, **-lock**, and **-unlock** options.

## Adding or Changing Files in a WAIS Database

Suppose you would like to add more documents to your WAIS database without having to reindex the entire database. First, add the new documents to your data set. Then update the WAIS index by using the optional **-append** switch to **waisindex**. The command syntax is:

**waisparse** *filespec* | **waisindex -d** *existing-index* **-append**

where `filespec` specifies the files to be considered and `existing-index` specifies the pathname of the existing index files. A command line of this form tells **waisparse** to parse the list of files specified by `filespec`. It then pipes the resulting stream of documents to **waisindex**, which modifies the existing index files. Note that any files included in `filelist` that have already been indexed in `existing-`

*index* and that have not been modified since, are not reindexed. Only the members of filelist that are new or modified files are parsed and indexed.

Let's take an example.

```
% waisparse data/books/*.txt | waisindex -d indexes/books -append
```

This command line updates the index files in the **indexes/books** directory to account for any **.txt** files that are new or changed since the index files where last modified.

If you need to append to your database on a periodic basis, this procedure can be wrapped into a shell script and the UNIX **cron** facility can be used to periodically execute the script.

**Caution:** During an append operation, **waisindex** creates a temporary index file and then merges it with the original before overwriting the original index files. Be prepared for this sequence to require approximately twice the disk space of the original index file.

## Removing Files from a WAIS Database

Suppose documents in your data set become obsolete from time to time. If you need to remove or exclude documents from your database, you can update the WAIS index without suspending your users' service. First, use the **waisdelete** command as described in this section to strike the obsolete documents from the WAIS index. Then, you may optionally delete the files or portions of files containing the documents.

The waisdelete command does not delete data, nor does it compress the index files. It simply "turns off" specific documents so that they are invisible to the **waisserver** program. The **waisdelete** syntax is:

**waisdelete -d** *dataindex* **-x** *docspec* [optional: *startbyte len*]
**-x** *docspec* [optional: *startbyte len*]
...

Here, *dataindex* is the pathname of the directory containing the current WAIS index. The *docspec* argument is one of two things, depending on whether your data is file-based or DBMS-based.

If your data is file-based, the **waisdelete -x** switch must be followed by the full pathname of the file containing the document to be removed. If a file can contain more than one document, you must also specify the

starting byte of the document within its file (the `startbyte` parameter), and the length of the document in bytes (the `len` parameter).

If your data is DBMS-based, the **-x** switch must be followed by only one parameter: the document key (doc-key). This is a unique identifier that has been assigned to the document by the custom parse format developed to handle your DBMS format. The assignment of doc-keys is site-specific and therefore keeping track of them is a WAIS administrator's responsibility. (For more information on DBMS-based data, see Chapter 6, "Customizing a WAIS System.")

In either case, one **-x** switch argument is needed for each document to be removed from the WAIS database.

The - option to waisdelete causes it to read its arguments from standard input. This is especially useful if you want to read deletion requests from another program. Also, if you have many deletions to make, the - option sidesteps the UNIX command line limit.

For example, in

```
% cat delete-args-file | waisdelete -d dataindex -
```

the **delete-args-file** is the name of a file containing lines of the form shown below.

```
-x
doc-id-1
start-byte
len
-x
doc-id-2
start-byte
len
...
```

Notice that each argument must be placed on a separate line. Thus, in a file-based system, **delete-args-file** might look like this:

```
-x
/usr/martin/mail-archive
4041968
24601
-x
/usr/martin/mail-archive
4121861
94309
```

Here we see Martin has kept mail archives with multiple documents in each file and provided the offsets and lengths of the ones to be ignored by the WAIS server.

## Merging and Replacing WAIS Databases

Two other waisindex options that help manage changing databases are **-merge** and **-replacewith**.

The **-merge** option joins two sets of index files. It takes two arguments, the pathnames of the index directories to be combined. For example,

```
% waisindex -d final-db -merge db1 db2 -finalize
```

will merge the wais/indexes/db1 and wais/indexes/db2 index files into wais/indexes/final-db. Notice that the destination index directory is specified as an argument to the -d switch. In addition, the -finalize switch is required when using -merge; it makes the resulting, merged index searchable.

The **-replacewith** option is used to replace the database currently being used with a new database. The syntax is:

**waisindex -d** *current-db-index* **-replacewith** *new-db-index*

As an example of using these two options, consider a situation in which you need to make massive updates to a database because a whole new set of informational files has been obtained. You could use the **-append** option and simply issue the command

```
% waisparse *.txt | waisindex -d db/index -append
```

But suppose there are so many files that you need to process them in batches.

This command sequence accomplishes the same thing:

```
% waisparse [A-Z]*.txt | waisindex -d db/tmp1
% waisparse [a-z]*.txt | waisindex -d db/tmp2
% waisindex -d db/update -merge db/tmp1 db/tmp2
% waisindex -d db/merge -merge db/update db/index
% waisindex -d db/index -replacewith db/merge
```

Two temporary indexes are made by parsing and then indexing the new data files in two portions. The temporary indexes are then merged—first with one another and then with the original index. Finally, the original index is replaced with the newly updated one.

## Locking a WAIS Database

When waisindex is executing, it is careful to make sure that: 1) no two processes are modifying the WAIS index at the same time, and 2) users are not accessing the index files while they are being overwritten.

The -append, -merge, and -replacewith options to waisindex as well as the waisdelete command automatically take care of these checks. If another WAIS process had started modifying the same database before your process did, your WAIS process will wait until that other process has finished. It will write out messages that look like:

**% 3540: 11: Aug 25 11:37:49 1994: -2: this database is already being updated waiting for lock on database, still trying... (^C to abort)**

until it has permission to start appending, or until you kill it. The locking mechanism used to accomplish this is available to you with the -lock and -unlock options described below.

The -lock options to waisindex provides an explicit means of locking out users and processes at times when reading or writing the WAIS database files could compromise their validity or yield invalid results. Each lock option is described below.

**-lock read**

The read lock is used while reading the WAIS index. It prevents any other process from modify the index. The read lock prohibits write locks from being placed on the database. However, multiple read locks can be put on the same database.

**-lock update**

The update lock is used while updating the WAIS index. It prevents any other process from attempting to simultaneously modifying the index. The update lock prohibits write locks and other update locks from being placed on the database. However it allows read locks to be placed by other processes.

**-lock write**

The write lock is used while writing to the WAIS index. It prevents any other process from accessing the index. This is an exclusive lock: the write lock prohibits any other locks from being placed by other processes.

**-unlock**

The unlock option is used to kill update and write locks. For example, you need to remove the locks left behind when indexing processes have died.

The syntax for a lock option is:

**waisindex -lock** *read-write-update* **-d** *index-pathname*

where *read-write-update* specifies the kind of lock and *index-pathname* is the pathname of the WAIS index.

When you invoke any of the -lock options, simply type the command line followed by a carriage return. To end the lock, type another carriage return.

You can run a lock in the background or in another shell while you're reading from or writing to the WAIS index. Simply add the ampersand (&) character to the end of the command line. When you are done with the operation you want protected by the lock, simply move the lock process to the foreground: at the UNIX prompt, type fg followed by a percent sign (%) and the background process's ID. Then enter a carriage return to end the lock.

As an example, suppose you need to copy a WAIS index in order to move it from one machine or directory to another. The command

**% waisindex -lock read -d /wais/indexes/currentdb &**

locks out modifications to the /currentdb files and yeilds this response:

**Lock established. It is only valid for the duration of this process.Press return to continue.**
**[1] + Stopped (tty input) waisindex -d /wais/indexes/currentdb -lock read**

The line beginning with [1] indicates that the lock has been assigned background process ID of 1 and reiterates what it is doing. When the UNIX prompt appears, you can copy the index files, secure in the knowledge that they can not change until you are done.

**% cp -r /wais/indexes/currentdb /wais/indexes/olddb**

This line recursively copies the contents of /currentdb and all its subdirectories into /olddb. After this command completes, bring the lock process to the foreground. At the UNIX prompt, type

**% fg %1**

UNIX responds:

```
% waisindex -d /wais/indexes/currentdb -lock read
```

indicating that the lock process has been brought back into the foreground. Now enter a carriage return to end the lock on the index files.

Consider next a situation requiring the **-unlock** option. The syntax for the -unlock option is simply:

**waisindex -d** *index-pathname* **-unlock**

where *index-pathname* is the pathname of your WAIS index files.

Suppose a waisindex process has died while doing an incremental indexing -append operation . Perhaps there was a power outage. After the power is reinstated, you unlock the database and start again.

First you invoke an append.

```
% waisparse /wais/data/*.txt | waisindex -d /disk/db1 -append
```

But the power goes out before the process has completed. When the power resumes, you once again invoke an append. But append left its lock behind:

```
3540: 11: Aug 25 11:37:49 1994: -2: this database is already being updated
waiting for lock on database, still trying... (^C to abort)
waiting for lock on database, still trying... (^C to abort)
waiting for lock on database, still trying... (^C to abort)
```

You abort and issue the unlock command.

```
% waisindex -d /disk/db1 -unlock
```

Unix responds

```
unlocked
```

Now, incremental indexing can succeed.

```
% waisparse /wais/data/*.txt | waisindex -d /disk/db1 -append

4438: 0: Aug 25 12:31:29 1994: 7: parsing /wais/data/text1.txt
4439: 0: Aug 25 12:31:29 1994: 100: updating /wais/data/text1.txt
...
```

The waisparse and waisindex programs execute while outputting their normal status messages.

# 5

# Setting Up a WAIS Server

This chapter describes how to bring up a WAIS server. WAIS is an extremely flexible system. Therefore, different command options are used depending on the nature of the database you are serving. It is important to decide which options are best suited to your particular database.

Before using the commands explained here, read through this chapter and decide which of the modes of operation described you will use. Also, look through the section entitled "Server Customizations," in Chapter 6, "Customizing a WAIS System." You may also want to incorporate some of advanced features described there.

## What is a WAIS Server?

The waisserver is a process that runs on a machine containing one or more WAIS databases, and services requests from client processes. The operation of the server is very simple. First a WAIS client initiates a request to a server machine. Next, the server machine responds by creating a new server process to handle the client's request.

Once created, the server process reads the client's question, performs a search on the requested databases, and returns a relevance-ranked list of document headlines to the client. If the user wants to view one of the documents, a retrieval request is sent from the client to the server. The server verifies that the document requested is in the database, and if so, retrieves the data from the original file that was indexed. Finally, when the client has completed all requests from the database, the server process terminates.

# Preparing for Server Set-Up

Before setting up your WAIS server, you will need to decide on the items described below. Where appropriate, we have also included our recommendations for how to approach each item.

- The port number on which to run the server. This is a number that is agreed upon between the WAIS server and WAIS client programs that wish to communicate with it. It is common practice for public WAIS servers to use port number 210.
- The root directory where the indexes of your server's databases are located. The suggested location is in a directory named /wais/indexes.
- The name and location of the log files. The recommended location is in the file named /wais/logs/server.log.
- The user that will run the server. Typically, this is the superuser, the user named root.
- The mode the server will run in, either standalone mode or daemon mode. In practice, standalone mode is only used for debugging purposes, and daemon mode is recommended for long term use. Setting up the server for standalone and daemon mode are described in this chapter.
- The Internet addresses (IP address) of those machines having access to your server's databases. Setting up restricted access lists for these machines is detailed later in this chapter.

To set up your WAIS server, we begin first by describing how to run your server in standalone mode. Once you feel confident that the server has been correctly set up in standalone mode, we then present how to set it up in daemon mode for long-term usage. In Chapter 6, we describe how to set up an access security list to provide restricted access to each of your databases.

# Standalone vs. Daemon Mode

For each new client requesting services, the responsibility for creating a new server process depends on whether the server is being run in standalone or daemon mode. In standalone mode, a WAIS server process is always in existence and continually running in the

background. For each new client requesting service, this process forks off a new child server process to handle the client.

In daemon mode, the Internet network daemon, **inetd**, is responsible for creating a new server process for each new client requesting service. The **inetd** daemon manages many network services according to the configuration specified by the /etc/inetd.conf file. The **inetd** daemon runs in the background and manages client requests by spawning off a child server process to service these requests.

We suggest that you initially try running the server in standalone mode because the server is much easier to test in standalone mode than daemon mode. In standalone mode, the server is started directly from the shell. This output allows you to easily detect if anything goes wrong with the server. On the other hand, in daemon mode there is much less feedback in the event of an error, and it is thus more difficult to debug.

Once the server has been tested in standalone mode, the WAIS server can then be set up to run as an new network service of the **inetd** daemon. Running the server in daemon mode eliminates the need to have a separate server process to spawn off child servers for each new client process. Instead, the **inetd** daemon services these requests itself by forking new child servers, one for each new client.

# Standalone Mode

To run the WAIS server in standalone mode, you can invoke the **waisserver** program either at a UNIX prompt, or through a shell script. To run the program, use the following command:

**waisserver -p** *port* **-d** *directory*

where *port* is the port number through which the server communicates with the client, and *directory* is the root location of the index files.

If the port number is less than 1024, the command must be run as superuser. It is common practice to use port 210 for public WAIS servers, but for testing purposes, you may want to choose a different port number that does not require you to be superuser. If you choose to select a port number other than 210, first check the /etc/services file to ensure that this port number is not being used by any other network service.

An example usage,

```
% waisserver -p 8000 -d /wais/indexes &
1: 0: Nov 2 05:09:43 1994: 100: running server WAIS Inc
```

selects port number 8000. The **&** at the end of the command line tells the UNIX shell to run the command in the background. This allows you to type other commands at the prompt while the server process is running. Once you have started the server, check to see that the server process has started:

```
% ps -auxww | grep waisserver
margaret  3207  0.0 2.2 2.16M  360K p4 S    0:00 waisserver -p...
margaret  3212  0.0 1.3 1.52M  208K p4 S    0:00 grep waisserver
```

In this example, the server's process identification (PID) number is 3207. You may want to keep track of the PID number in case you need to kill the process.

And for the final step in the testing process, you may want to run **waislookup** to verify that the data in each of your databases is accessible through the server. This is described in the section, Testing Your Server.

Once you have completed and verified that the server runs in standalone mode, you may then want to kill your server process and try running the server in daemon mode. For the **kill** command, use the PID number that you recorded from running the **ps** command.

```
% kill 3207
2: Terminated waisserver -p 8000 -d /wais/indexes...
```

If you decide to continue to run the server in standalone mode, keep in mind that the server will need to be restarted every time the machine is rebooted. This can be facilitated by putting the command in your /etc/rc.local file — or the equivalent local startup file.

# Daemon Mode

To run the server in daemon mode, you must log on as superuser and modify two files: 1) the /etc/inetd.conf and 2) the /etc/services file.

A sample entry for setting up the server in the **/etc/inetd.conf** file follows:

**z3950 stream tcp nowait root /wais/current/bin/waisserver waisserver.d -e /wais/logs/server.log -d /wais/indexes/my-db**

The entry must be specified as a single line in the **/etc/inetd.conf** file (no carriage returns). Each item in this entry is described below:

**z3950**

The service name is z3950. The service is typically named after the protocol used, but it can be any alphanumeric symbol. The same service name must be used in the /etc/services file described later in this section.

**stream**

The socket type is stream. A socket is a bi-directional connection between processes. A stream is a continuous flow of data. This means that other processes contacting the server will expect to send and receive data streams.

**tcp**

The transport protocol is TCP. This is the fundamental Internet protocol.

**nowait**

After starting up a new server process, nowait specifies that the **inetd** daemon should not wait for it to complete before servicing another request from a new client.

**root**

The user running the server program is root.

**/wais/current/bin/waisserver**

This is the name of the command to run to invoke the server. This should be set to the complete pathname of the **waisserver** executable at your site.

**waisserver.d**

When a server program is activated by the **inetd** daemon, the name of the server program is **waisserver.d**. This is a required field an dmust not be changed.

**-e /wais/logs/server.log**

The **-e** argument to the **waisserver** program specifies the filename for where the server should log its information. See Chapter 7, Monitoring WAIS Usage, for more information on the log file and generating WAIS Usage Reports.

**-d /wais/indexes/my-db**

The **-d** switch to the **waisserver** program specifies the directory containing the database index files.

Additional switches may also be specified to the **waisserver** program by appending them to the **inetd** entry.

Be careful, however, because some inetd programs will only allow a limited number of arguments in the daemon command. To get around this restriction, use the -args option to waisserver. This option takes one argument, the name of a file containing the optional switch arguments to waisserver. For example, to use the -args option in the inetd.conf daemon command described above, we would write

**z3950 stream tcp nowait root /wais/current/bin/waisserver waisserver.d -args /wais/indexes/my-db/argsfile**

and the argsfile contents would look like this:

**-e**
**/wais/logs/server.log**
**-d**
**/wais/indexes/my-db**

Notice that an args file must have only one symbol per line.

The next step required for daemon mode setup is to add an entry to the **/etc/services** file. The new entry to this file looks like:

**z3950 210/tcp # Wide Area Information Server (WAIS)**

This entry tells the **inetd** daemon what port number and protocol to use for the service defined in the **/etc/inetd.conf** file. Each item in the services file entry is described below.

**z3950**

This must be the same service name as specified in the /etc/inetd.conf file.

**210**

The port number that a client process should use to talk to the server is port 210. This is the standard port number for public WAIS servers.

**tcp**

The protocol to use for communicating is **tcp**.

**# Wide Area Information Server (WAIS)**

Anything after a # character is interpreted as a comment. This comment lets other system administrators know about this service and corresponding port number.

Whenever you modify either the **/etc/inetd.conf** file or the **/etc/services** file, you need to restart the inetd daemon so your modifications will take effect, . This can be done by rebooting, or by finding the **inetd** daemon PID number with **ps** and sending a HUP signal to the process.

```
% ps -auxww | grep inetd
root   136 80 330 17 ?    S  Nov 2 0:04 /usr/sbin/inetd -s
morris 2734 12 135 95 pts/2 S   00:31:45 grep inetd
% kill -HUP 136
```

The line kill -HUP 136 sounds drastic, but it simply tells process number 136 to reread its inetd configuration file.

At this point, you may also want to update your NIS or NIS+ server.

Note that, if you use the -args option to waisserver, you can alter the daemon's behavior simply by modify the argument file: you need not edit the inetd.conf file, nor login as superuser, nor send a kill -HUP signal to make your changes take effect. Instead, each time a client newly connects to the Z3950 service, inetd will start up a new waisserver process which will reread the args file.

Once the inetd daemon is using your modified **/etc/inetd.conf** and **/etc/services** files, you are ready to test the WAIS server in daemon mode. As in the standalone mode, try performing a search and retrieval using the **waislookup** client.

When the server is running in daemon mode, the server is automatically started whenever a client connects to it. You do not need to do anything special, even after rebooting the machine.

# Testing Your Server

The waislookup program can be used for testing and troubleshooting WAIS servers. To test a database using waislookup, use the following command line:

**waislookup -d** *database-name@server:port*

With this command, you should be able to successfully run queries. For more information about waislookup, see Chapter 8 and Chapter 9.

If you are not able to run queries, there are several possible causes. Check the **inetd** entry, making sure that the files exist, permissions are correctly set, and the user you selected exists. Check that **inetd** has been restarted, and reboot if necessary. Make sure the machine you are using is capable of communicating with the machine the server is running on. Finally, try running the server in standalone mode, with the same arguments you used in the inetd.conf.

Once you get this working, try connecting using the Macintosh or other WAIS clients.

# 6

# Customizing a WAIS System

Four categories of customization are explained in this chapter.

- **Meta-Document Customizations**

  Meta-documents are documents that help a WAIS client user use your database. The WAIS Administrator can create various meta-documents in files that are returned to end users at appropriate times to tell them about the unique aspects of a particular WAIS database. For example, the index.hlp file provides on-line help.

- **Indexing Customizations**

  When building an index for your WAIS database, there are a number of optional features you can use. In addition, the nature of the data may require that you undertake some customization, such as writing a custom parse format.

- **Server Customizations**

  When bringing up the WAIS server, a variety of optional features are available, each of which will change the operation of the server to some degree. In certain cases, the nature of your database requires that you add customizations to the server. For example, if you are serving data that resides in a non-WAIS database, writing a custom filter program is necessary.

- **WAISgate**

  The WAISgate product provides a gateway between the World-Wide Web and your WAIS server. By using WAISgate you can make your WAIS database accessible from the World-Wide Web. Similarly, WAISgate can make Web pages more accessible by providing Web clients with WAIS searching capabilities.

# Meta-Document Customizations

## On-Line Help and Notice Files

There are two ASCII text files you can create to keep your WAIS database users well informed:

1) The custom help file named **index.hlp**

2) The custom notices file named **index.not**

If you put these files in your index directory, the WAISserver will, at the appropriate times, send their contents to the client process for display to end users.

### Help .i.help file;

A WAIS client user can request help by typing the single word **help** as the query, or by issuing an empty query.

As described in Chapter 4, "Building a WAIS Database," the database description in the *database*.src file is returned by default (unless waisindex was run with the -nosrc option). However, if the file **index.hlp** is present in the WAIS index directory, its contents will be returned to the client for display to the user instead.

In addition, the .cat file is returned to the client user in response to a help request (unless waisindex was run with the -nocat option).

Use the index.hlp file to provide your users with information about the contents of the database, how to search it, its price structure, and whatever else you know they will find helpful. WAIS, Inc. provides a sample text file describing how to perform WAIS searches. You may want to include some or all of this text in your index.hlp file. Its pathname is wais/curent/doc/howtosearch.txt.

### Notices

A special notice can be displayed at the beginning of every results list that the waisserver returns in response to a query. If you have a special legal notice or another notice you'd like conveyed in this manner, simply put them in a file named **index.not** in the WAIS index directory.

# Indexing Customizations

The optional features described in this section will affect the results of running waisparse and waisindex to build a WAIS index.

Before building your index, look through both Chapter 4, "Building a WAIS Database," and this section. Decide which optional features, if any, you will use. Then, return to Chapter 4 and follow the instructions there, supplemented by selected portions of the instructions in this section.

## Stopword List

A stopword is a frequently used word that, when encountered in a user question, is ignored. For example, since the word "the" commonly appears throughout the English language, it is typically regarded as a stopword. You can specify a customized file of stopwords as an optional argument to the **waisindex** program using the **-sw** switch.

**waisindex -d** *database-name* **-sw** *filename*

where *filename* is the pathname of the file containing your customized list of stopwords. Each stopword in this list is separated by a carriage return. A different stopword list may be specially customized for each database you maintain.

If a stopword list is not specified, the WAIS indexer uses a built-in list of approximately 300 stopwords. This default list is provided with the installation in the file /wais/current/util/stopwords.txt.

The stopword list is also included in Appendix C. You may use this file as a template for creating your own stopword list. If you do not want any stopwords, you can create one empty stopword file, or use /dev/null, which is the name of a special empty file on most UNIX systems.

> **Note:** If you plan to merge databases or to use a multi-database, be careful to use the same stopwords for all the databases involved.

## Stemming

Stemming is a technique used to automatically derive variations of a queried word. These variations are then used as part of the search. If a

question contains the word "skate", for example, stemming is used to find documents that may also include "skates", "skated", and "skating". Two types of stemming are supported: Plural and Porter stemming. Plural stemming attempts to determine the plural form of a word. Porter stemming attempts to find the real base, or stem, of a word and derive any possible alternate variations.

You can specify the stemming algorithm as an optional argument to the **waisindex** program with the **-stem** switch as follows:

**waisindex -d** *database-name* **-stem** *stemmer-name*

where *database-name* is the pathname for the database directory, and *stemmer-name* is one of **plural** or **porter**. A different stemming algorithm may be used for each database you maintain. If this option is not specified, the default is to use no stemming. Example uses include:

```
% waisindex -d /wais/indexes/resumes -stem plural
% waisindex -d /wais/indexes/jargon -stem porter
```

> **Note:** If you plan to merge databases or to use a multi-database, be careful to use the same stemmer for all the databases involved.

## Custom Parser Toolkit

There are a large number of built-in parse formats shipped with the WAISserver software. To see a list of the parse formats supported in the current release, type

**% waisparse**

with no arguments at the UNIX prompt. To see a list of parse formats contributed by customers and the freeware community, type

**% waisparse -c**

If your data type is not covered by a provided parse format, you will need to write a custom parse format to enable the waisparse program to distinguish the headline, body, words, and possibly the fields represented by your data format. Custom parse formats installed at your site are listed if you type

**% waisparse -l**

The Custom Parser Toolkit is a well documented C program that includes pseudo code for all the customized functions you may need to write to define a new parse format. The toolkit is included with WAISserver software, Version 2.0 or higher. For instructions on how to use the

Custom Parser Toolkit, see the file **wais/current/parser-toolkit/parser-manual.txt**.

Alternatively, WAIS Inc. can provide consulting services to develop and maintain customized parse formats to recognize your data. Please contact WAIS Inc. if you would like to purchase custom programming services on a contract basis.

## Tagging Database Fields

For data sets containing documents that are structured in a semi-regular format, the regular portions of the documents can be tagged as fields during index creation. A query can then restrict the search to given field values. Performing a restricted search based on the value of a field or set of fields is called *fielded search*. In its simplest form, a query containing a fielded search is specified as follows:

*field-name* = *field-value*

A query of this form tells the WAIS server to only search for documents whose field, specified by `field-name`, contains the value specified by `field-value`. The WAIS parser can support up to 254 unique fields.

Within each field, the WAIS search engine carries out a natural language search. Note however that fielded searches are not required when querying a fielded database. Unlike a DBMS, the WAIS search engine can perform searches over data stored in any format using simple natural language, literal, and boolean queries.

If your database contains fields, use the Custom Parser Toolkit described above to help you create a custom parse format that recognizes the fields. Note that it is possible to tag more than one field from the same line in a data file.

Once you have created a parse format to identify the fields in your database, be sure to list the field names and their allowable values in the textual database description you add to the *database*.src file. That way, users browsing the directory of servers will know what fields they can use when searching the database.

The **mail-or-rmail** parse format is a good example of a parse format in which fields are tagged. Using this parse format, the WAIS parser detects the **To** and CC fields, the **From** and **Sender** fields, the **Subject** field, and the **Date** field. As an example of a query using fielded search,

subject = tee shirts AND from = david

would return documents that contain both a **Subject** field value of tee shirts and a **From** field value of **david**, where **AND** is a boolean operator for set intersection.

For the date field, a range of dates may be specified, using the syntax

**data** *comparison-operator value*

where the `comparison-operator` may be one of the following symbols:

| Comparison Operator | Meaning |
|---|---|
| > | greater than |
| < | less than |
| >= | greater than or equal to |
| <= | less than or equal to |
| = | equal to |

**Comparison Operators for Range Queries**

The `value` argument in a date range query may be a date specification in one of the following formats:

| | |
|---|---|
| 7-4-94 | 7/4/94 |
| 07/04/94 | 7.4.94 |
| 07-04-94 | 7-4-1994 |
| today | yesterday |

**Date Formats for Range Queries**

If the comparison-operator is =, then a range may be specified using the TO operator, as in

**date = 1/6/96 TO 6/6/96**

Here, both ends of the date range are inclusively specified.

Date ranges may be searched in the manner described above for WAIS databases using the **mail-digest**, **mail-or-rmail**, or **netnews** parse

formats. These three parse formats are built-in parse formats that support fielded searches.

Range searches using the same comparison operators and **TO** syntax as described above for dates are supported for any positive integer field values. To enable positive integer range searches over your database, use the Custom Parser Toolkit to define a parse format that includes the appropriate integer field definitions.

# Server Customizations

The optional features described in this section will affect the results of running waisserver to bring up your WAIS server process.

Before bringing up your server process, look through both Chapter 5, "Setting Up a WAIS Server," and this section. Decide which optional features, if any, you will use. Then, return to Chapter 5 and follow the instructions there, supplemented by selected portions of the instructions in this section.

## Setting Up Security

The WAISserver software uses an access-list security system to limit client access to WAIS databases. Before processing a client's request, the server checks to see if the requesting client has access to the requested database. It does this by checking an access list maintained for each database. The access list tells the server the legal client machines that have access to the database.

For any given database, the access list is a file with a **.acc** extension and is located in the same directory as the indexes of that database. For example, if the pathname of your database index directory is

**/wais/indexes/mydatabase**

as specified by the **-d** switch to the **waisindex** program, then the pathname of the access security list file for this database should be:

**/wais/indexes/mydatabase/index.acc**

If this file is nonexistent, any client can access the information in the database. If this file does exist, then only those clients whose Internet address is listed in the file will be allowed access the database. All others will get an error message, suggesting the user contact the database maintainer.

The .acc file consists of the Internet address (IP address) of certified client machines, such as:

```
IP:+:192.216.46.104
```

The "IP:+:" must appear exactly at the beginning of a line, followed by the IP address in Internet dot (.) notation. Comments can be entered into the file by using a hash mark (#):

```
#This is a comment
IP:+:192.216.46.104
#This is another comment
IP:+:192.216.46.104  # yet another comment
```

Asterisks can be used in the IP address to replace for any of the 4 component numbers. For example, the entry,

```
IP:+:192.216.46.*
```

allows access to machines with IP addresses 192.216.46.0 through 192.216.46.255, and the entry,

```
IP:+:192.216.*.*
```

gives access to machines with IP address from 192.216.0.0 to 192.216.255.255.

## Configuration Files

A WAIS configuration file defines various run-time parameters that the waisserver program uses when processing requests against a WAIS database. Configuration files are auxiliary files optionally added by you, the WAIS administrator. There are two types of WAIS configuration files: local and global.

You may define a local, or database-specific, configuration file by naming it **index.wc** and placing it in the index directory of the database you want it to affect.

In addition, you may define a global configuration file that affects any databases that are served by your WAIS server. There are two ways to define a global configuration file.

1. Create the configuration file you want as the global default. Give it any name you like. When you call waisserver, use the -config option, which takes one argument: the pathname of the global configuration file. For example, the command

**waisserver -d /wais/indexes -config /wais/current.wc**

invokes waisserver using /wais/indexes as the base directory and /wais/current.wc as the default configuration file.

2. Alternatively, create the configuration file you want as the global default and put it in one of the following files:

**wais.wc** (directory from which the server is executed; / for inetd )
**/usr/local/wais/wais.wc**
**/usr/local/wais/etc/wais.wc**
**/etc/wais.wc**

If waisserver was not invoked with the -config option, the server process searches for a global configuration file by checking each of the pathnames listed above in the order shown.

A configuration file has strict formatting requirements. Each line is a declaration composed of a symbol followed by zero or more values, as in

*symbol value1 value2 value3 ...*

where *symbol* is a case insensitive copy of one of the parameters listed below, and *value* must be enclosed in double quotes if it contains any whitespace. Comments may be included in a configuration file on lines that each begin with a hash (#) mark.

Each available configuration parameter is listed and described below. Note that it is not necessary that every available configuration parameter be included in your configuration file.

**MAINTAINER**

The WAIS Administrator's email address. Used in the Query Report, which ends with "For more information send email to <maintainer>." Defaults to "info@wais.com".

**HELP-HEADLINE**

Headline for user help file, **index.hlp.** Defaults to "Help on database: *name-of-database.*"

**NOTE-HEADLINE**

Headline for user notices file, **index.not.** Defaults to "Information on database: *name-of-database.*"

**QR-HEADLINE**

Headline for Query Report. Defaults to "Query Report for this Search".

**SRC-HEADLINE**

Headline for Source (**.src**) file. Defaults to "Information on database: *name-of-database*".

**CATALOG-HELP-HEADLINE**

Headline for Catalog (**.cat**) file when user types "help". Defaults to "Catalog for database: *name-of-database*".

**CATALOG-NO-HITS-HEADLINE**

Headline for Catalog (**.cat**) file when query yields no documents. Defaults to "Search produced no result. Here's the Catalog for database: *name-of-database*".

> **Note:** In each of the headline declarations ( help, note, qr, src, catalog-help, and catalog-no-hits), the symbol percent-s (%s) may be placed in any one place and the name of the database will be substituted, e.g., setting the help headline to "How to use %s" will result in "How to use *name-of-database*".

**METADOC-AT-BOTTOM**

(no value) If this symbol appears in a WAIS configuration file then, for the affected databases, the catalog and help file (or **.src**), will be at the bottom rather than at the top of the results list if the user requests help.

**DBType**

Database type declaration. Defaults to WAIS (the standard default WAIS database). This parameter is meaningful in database-specific configuration files (index.wc). Following the parameter name must be another symbol specifying the database type. Currently there is one choice other than the default: MULTI. A multi-database is composed of two or more WAIS databases. If a DBType MULTI declaration appears in a configuration file, at least one DBS declaration (shown below) must also appear in the same file. For maintainability, a DBS declaration generally follows immediately after a DBType declaration.

**DBS**

Multi-database member declaration. Only used in conjunction with DBType Multi declaration. Must each begin with the symbol DBS followed by any number of full pathnames indicating which databases comprise a multi-databases. For a complete discussion

of muti-databases, see the section entitled "Multi-Databases," in this chapter.

**FILTER**

Declares filter programs, which retrieve and optionally modify data. Defaults to nothing, that is, no filter used. Takes multiple values in sets of three. Syntax for the whole line is thus:

**FILTER** *display-typeA display-typeB filterprogram*

Within each filter specification, the first two values are display formats. The third value is either the full pathname for a filter program or a series of UNIX shell commands: if there is any whitespace, the shell commands must be enclosed in double quotes. Multiple filter declarations may be included in a single configuration file.

For a complete discussion of filter programs, see the section entitled "Creating and Using Filters," further on in this chapter. .

**LICENSE**

A unique alphanumeric value provided by WAIS, Inc. to specify wais licenses. You will be told what value to give this parameter in order to turn on your WAISserver software. Multiple values per line and multiple lines are allowed. Do not change the value you are given, it encodes the date on which the software license expires.

Before serving any data, a server process first checks for configuration files. If you have created a global configuration file, its parameter values are the defaults. For any particular database, they are overridden by the parameter values given in the local index.wc file, if there is one.

## Timeouts for Idle Clients

The **waisserver -t** option defines how long the server process will maintain a connection with an idle client process.

By default, when a client has been idle for 10 hours, the server closes its connection. You can change this timeout with the **-t** option. The syntax is:

**waisserver -t** *how-long other-waisserver-options*

where the parameter *how-long* is the number of seconds of client idle time allowed before the connection is cut. For example, a setting of **-t 3600** would wait only 1 hour.

## Multi-Databases

The WAISserver software is capable of serving multiple databases as if they were one big database. A database configured in this manner is called a multi-database.

The benefits of a multi-database are several:

- **size and speed**

  A very large set of data is often more easily managed if split up between two or more directories — possibly on separate disks. In some cases, this becomes mandatory because the WAIS index files (especially .inv and .dic) approach or overflow the file limit for the UNIX implementation you are using. In other cases, splitting a database into two or more portions is desirable because it cuts down on indexing time and space requirements when updating or modification is required. Even incremental indexing can bog down your server machine when processing a huge database.

- **flexibility**

  To WAIS client users a multi-database appears to be a single database. However, depending on how it is set up, a multi-database can offer users the greater flexibility of searching each component database separately or searching them all as a multi-database. The WAIS administrator can choose whether or not to publicize the component databases by creating and entering their .src files in a directory of servers. Unless you inform them, users will assume the multi-database is simply another WAIS database.

- **accuracy**

  The WAISserver multi-database mechanism ensures that document relevance scoring is merged across the component database searches. Although a client process can search multiple databases without needing to have them defined as multi-databases, accurate score merging is not done by WAIS clients.

A multi-database is easily set up simply by creating a directory for the multi-database and putting two files in it: 1) a WAIS configuration file named index.wc and 2) a source file named *database*.src.

For example, suppose you want to set up a multi-database named /wais/indexes/fish-and-fowl, which searches two other databases named /wais/indexes/fish and /wais/indexes/birds. First create an index directory for the multi-database. From within the indexes directory, type

```
% mkdir fish-and-fowl
```

Then, make the configuration file /wais/dbs/fish-and-fowl/index.wc and put the following two lines in it:

```
DBType MULTI
DBS /wais/dbs/fish /wais/dbs/birds
```

When the server receives a request to search the fish-and-fowl database, it will look in the configuration file and discover that this is a multi-database. It will then perform searches over the two component databases and merge the results appropriately.

To publicize the multi-database, you need to make a *database*.src file for it and then add that to any directory of servers in which you want it included.

As explained in Chapter 4, "Building a WAIS Database," an .src file is automatically generated when you build an index for a database. Since you do not build an index for a multi-database, you need to make its .src file "by hand." This is easily done. Below is an example .src file. It is nothing more than a data structure with named fields. Each field name is preceded by a colon and the field values are placed after each field name. Sample values are included in the example.

```
(:source
  :version 3
  :ip-address "129.362.28.5"
  :ip-name "server-name:whereever.com"
  :tcp-port 210
  :database-name "/wais/indexes/fish-and-fowl"
  :maintainer "whoever@whereever.com"
  :description
"This is where you put a textual description of the
multi-database you are defining. Be sure to include a description of any tagged fields you
may have defined for the database. This is the information users will turn to to decide
whether or not to search your database."

)
```

Using your favorite editor, simply copy the :source structure into an empty file, replacing the sample values with valid values for your site, server, and multi-database. Notice that all slot values are either text strings surrounded by double quotes or integer values.

Now, follow the instructions in the Chapter 4 section entitled "The .src File and The Directory of Servers," to add your new multi-database to the local and/or public directory of servers. This will let WAIS client users know it is available.

## Creating and Using Filters

The WAISserver filtering feature provides a relatively simple way for you to enhance the functionality of the WAISserver when it retrieves documents from your database. The waisserver program can be configured to invoke a filter program in order to do customized processing during document retrieval. A WAIS filter program is a UNIX program written in any UNIX programming or scripting language, including C, perl, or csh. This section describe why you might want a custom filter and how to create one if you do.

Alternatively, WAIS Inc. can provide consulting services to develop and maintain customized filters to process your data. Please contact WAIS Inc. if you would like to purchase custom programming services on a contract basis.

### Uses of Filter Programs

The most basic use of a WAIS filter program is to modify data — specifically, to translate documents from one display format to another. Thus, if a document has been parsed using a certain display format but a client process requests that it be retrieved in a different format, a filter program can handle the translation in the course of retrieving the document.

Since a filter program acts as a document retrieval mechanism, it is not limited to modifying data. One popular use of filter programs is to retrieve documents from external or non-WAIS databases, as described in the next section of this chapter. Similarly, it is possible to a create a filter that retrieves documents from other WAIS databases, from other kinds of databases entirely, or from data that is brought together to create dynamic, virtual "documents" on the fly. Conceivably, a filter could be created to perform "agent-oriented" tasks with the document it

is retrieving. For example, it could fax it, email it, encrypt or decrypt it, or cache it for future retrieval.

Generally, a filter program will filter documents for a waisserver process that is handling a client program's retrieval request. In addition, a filter program is invoked by a waisserver process when relevance feedback is used during a search. Relevance feedback is a WAIS search feature that uses a whole document or portion of a document as a query. Thus, if a user's query has resulted in the retrieval of a particularly interesting document, she can say, in essence, "Find me more like this." During relevance feedback, the server must retrieve and read the contents of the chosen document and therefore the filter is triggered during this kind of retrieval also. For more information on relevance feedback, see the *Technical Description,* Version 2.0, Chapter 3, entitled "The WAIS Server."

The waisserver process retrieves a document by using the filename table in the WAIS index to determine the required document key. When no filters are in use, it then uses the document key to locate the document in the file system, reads the file, and either returns the requested data to the client process or uses it to perform relevance feedback.

Using a filter program inserts another step into this process: the waisserver process passes the document key to the filter program. The filter program then locates the document, reads the data, processes it in its own way, and returns a stream of data to the **waisserver** process. The waisserver process then either passes it on to the client process or uses it to perform relevance feedback.

## Creating Custom Filters

To activate the waisserver filter feature, you need to write one or more filter programs and put a filter declaration in the appropriate configuration file.

### Filter Declaration

A filter declaration is a line in a WAIS configuration file that defines what filter program to use on which documents in a WAIS database. The waisserver process invokes the filter program whenever the appropriate filter declaration conditions are met.

A filter declaration must be of the form:

**FILTER avail-display-type req-display-type filter-program**

The *avail-display-type* is the display type assigned to the document when it was parsed by waisparse. The *req-display-type* is the display type requested by the client process. The *filter-program* is the filter to call when retrieving a document under these circumstances.

A filter declaration must be placed in a WAIS configuration file as described in the section entitled "Configuration Files," earlier in this chapter. Once a filter program is declared, waisserver processes will automatically find and use it.

As an example, a local configuration file index.wc might contain these lines:

```
# convert images from tiff to gif
FILTER TIFF GIF tiff-to-gif-converter

# retrieve text files from the sql db
FILTER TEXT TEXT get-sql-data

# translate html files to text and dvi files to ps
FILTER HTML TEXT "dump-web | format-web"
```

The first declaration means that if a TIFF document is available, but a GIF document is requested, then the program tiff-to-gif-converter will be called to convert the display format. The second declaration means that if a TEXT file is available and a TEXT file is requested, the program get-sql-data will be called to retrieve the document from an SQL database. The third is a double declaration: it uses double quotes to surround a command sequence invoked when an HTML-tagged file needs to be converted to TEXT. A waisserver process always checks the configuration files to see whether it should use a filter.

### Filter Program API

Here we describe the filter program Application Programming Interface, or API, which specifies the programming interface between the filter program and the waisserver program. A filter program may be written in any UNIX programming or scripting language, such as C, Perl, or csh. It is called by a waisserver process with eight arguments. These identify which document to retrieve, specify the desired portion of the document to retrieve, and indicate any display format translation required. The filter program must locate and read the specified document or portion of the specified document from wherever it may

be stored, process it, and then write the resulting document to standard output.

The filter program is expected to handle the following eight arguments passed by the waisserver process.

**filter-api**

The version of the WAIS filter mechanism's API. If a version 2 WAIS filter API is trying to call a filter program expecting the version 1 API, for example, the filter program should write an error to the file and exit. The current version number may be found in the filter-codes.h file.

**doc-key**

This is the way that the WAISserver software refers to documents. If you are using GatherDocKey and FinishDocKey functions in a custom parse format, this is the doc-key assigned by the custom parser. Otherwise, the doc-key is the full pathname of the file containing the document.

**start-byte**

The document's location within the file, or, in the general case, the location of the document as an offset from the beginning of the data buffer designated by the doc-key.

**end-byte**

The last byte of the document within the retrieved file or data buffer. This value may be negative, indicating that the position of the data within its containing file or data storage unit can change. If *end-byte* is negative, the filter program should assume that the length of the data has changed since last indexed, and recheck its length.

**block-start-byte**

A number specifying a byte offset from the *start-byte*. The filter program should start retrieving a requested data block at this byte.

**block-len**

The size of the block to retrieve. If (*start-byte* + *block-start-byte* + *block-len* ) is greater than the length of the file or data buffer, then the filter program should shorten *block-len* accordingly.

**avail-display-type**
The display type in which the document is available. (From the first value in the filter declaration)

**requested-display-type**
The display type that the WAIS client requested. (From the second value in the filter declaration)

A sample call to a filter program from the command line looks as follows:

**% html2txt 1 /usr/ben/mypage.html 24601 26093 0 1492 HTML TEXT**

The filter program is called html2txt. It uses filter API version 1. In this case, the doc-key is a filepath. The 1492 bytes between 24601 and 26093 are requested and the document is translated from hyper-text markup language to ASCII text.

A filter program should always exit with an appropriate code. These codes are defined in the file filter-codes.h and described below. The code name is given first, then the code value, and finally a description.

**FILTER_RESULT_REQUEST_OK 0**
Return this error code when filter execution is returning successfully.

**FILTER_RESULT_REQUEST_OUT_OF_RANGE 1**
Return this error code when the *block-start-byte* argument is beyond the end of the document.

**FILTER_RESULT_BAD_VERSION 2**
Return this error code when the first argument is in error.

**FILTER_RESULT_TYPE_NOT_AVAIL 3**
Return this error code when the *requested-display-type* argument is not one this filter can produce.

**FILTER_RESULT_ACCESS_DENIED 4**
Return this error code when the filter program could not open the file or otherwise access the data associated with the doc-key.

**FILTER_RESULT_BAD_DOCKEY 5**
Return this error code when the doc-key argument is in error.

**FILTER_RESULT_INTERNAL_ERROR 6**
Return this error code when the filter program cannot proceed.

To demonstrate the input side of the filter program API, consider the following csh script, which just writes out all the arguments to standard output.

```
#!/bin/csh

while ($#argv)
 echo -n $argv[1] " "
 shift
end
echo ""
```

When you start developing a custom filter program, try using this script as your filter program at first. It will help you see how waisserver calls a filter program. Just type it into a file. Then create an index.wc file in your index directory and include a filter declaration in there. The filter is now installed. To test it, call waislookup or any other client. Initiate a search and then a retrieval request against the database into which you've installed the filter. In response to a retrieval request, you will see the arguments with which the filter program is called. The WAIS log also records the filter arguments.

## Serving External Database s

An external database is any non-WAIS database, such as an RDBMS, which has its own way of storing and retrieving data. An external database may reside on a remote machine, or it may be stored on a machine at your site.

The WAISserver software can serve a data set even if that data is controlled by another database management system. Providing your server has permission to access the data, you can enable them to search external data sources simply by adding a few customizations to your WAISserver software.

Many data repositories are kept in relational database management systems (RDBMS's) or stored in special ways that make them inaccessible toWAIS. To make such data accessible to WAIS, you need to write software that performs the tasks listed below.

**1A.** Dump the external database contents as a stream.

**1B.** Parse the external database contents using waisparse. This is done by creating a custom parse format as described in the section entitled "Creating Custom Parse Formats," earlier in this chapter.

2. Retrieve and optionally modify documents from the external database. This is done by creating a custom filter program as described in the section entitled "Creating Custom Filters," earlier in this chapter.

The first two tasks are necessary to provide the waisindex program with a stream of documents to index. The third task is necessary to enable the waisserver filtering mechanism to return external data to a client process.

To review, WAIS server integration with an external database is accomplished as depicted on the right-hand side of Figure 4: An index directory is build for the external database contents with the help of a custom dump program and a custom parse format. A filter declaration is included in the database-specific configuration file, index.wc, to point to a custom filter program that knows how to retrieve and possibly modify external data.

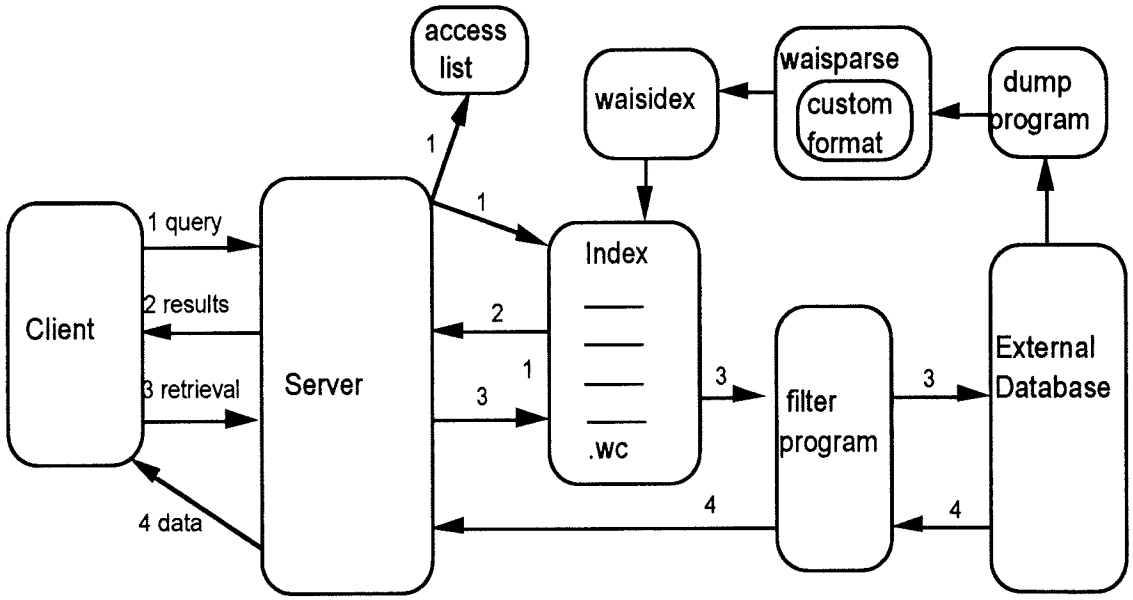



**Figure 4. Serving an External Database**

Once in place, using a WAIS server to access a non-WAIS database operates as depicted by the numbered arrows in Figure 4: First, the user issues a query (1). This causes a client process to send a search request to the WAIS server, which initiates a waisserver process. The waisserver process checks the access list to make sure the client's request is allowable. Assuming it is, the server process then reads the index.wc file to get any run-time parameters affecting this database.

Next, the server process consults the index files, which have been created with the help of a custom parse format, and returns (2) a search results list of headlines to the client. If the user asks to see one of the documents found, the client process sends a retrieval request (3) to the waisserver process. The waisserver process already knows from the filter declaration in the index.wc file that a filter program is needed to retrieve the document. It invokes the filter program, which retrieves the desired document, modifies the data or display format as needed, and returns (4) the data to the waisserver process. Finally, the waisserver process sends the data to the client process for display to the user.

Before undertaking the integration of an external database with your WAISserver software, we suggest you read the rest of this section and work through the example provided. Then, return to this point and follow the instructions in order.

## Step 1: Dumping External Data

The first task in making an external data source accessible to WAIS is to dump the data to a memory stream so that waisparse can process it. Note that, by using the + option, the waisparse program can take a stream of data as its input.

Confer with the external database administrator as needed and write a program that causes the external data source to output into a data stream the portion of it's data you want to serve. Most likely, this will be all the data the first time and only the new and changed data thereafter. Note: Typically, the stream is simply the terminal (stdout) which is then redirected using the UNIX pipe (|).

Once you have created a dump program to feed the external data to waisparse, write a custom parse format to decipher the data format.

## Step 2: Parsing External Data

To parse data from an external database, first be sure you have a complete description of the external data format. Then, follow the instructions in the Custom Parser Toolkit manual file to create a custom parse format for the external data. If you followed the default installation procedures, the custom parser toolkit documentation may be found in **wais/current/parser-toolkit/parser-manual.txt**.

For an overview of how to develop a custom parse format, see the section entitled "Creating Custom Parse Formats," earlier in this chapter. After writing a custom parse format and compiling waisparse with it, you can build a WAIS index for the external database.

To create a WAIS index for the external data, invoke waisparse and waisindex using the command syntax

*dump-data* | **waisparse -parse** *new-pformat* **+**
| **waisindex -d** *extern-index-path*

where *dump-data* invokes your program to dump the external data to standard out; *new-pformat* is the name of your custom parse format; and *extern-index-path* is the pathname of the directory into which you want the newly created WAIS index written. The + option to waisparse causes it to read its input from standard input rather than from a file.

## Step 3: Filtering External Data

Now that you have successfully parsed and indexed the external database, the next step is to enable the waisserver program to retrieve and optionally modify it for display to end users. This is done by writing a filter program.

Be sure you understand how to retrieve a document from the external database. Also, make sure you know in what display format the data is stored and what display format or formats your users are likely to request. With this information in hand, you can create a filter program for an external database. Simply follow the instructions in the section entitled "Creating Custom Filters," earlier in this chapter.

## External Database Example

You can experiment with the process of integrating an external database with WAIS by playing with the currency exchange example. This example is provided in the directory /wais/sample/currency-exchange. To follow along, cd to that directory now.

This example is a simple database that stores international currency exchange data. Although not all external databases are remotely located, we will pretend that the currency exchange database resides, not on your local server, but at the Thirty-Fourth National Bank, which keeps it up-to-date and makes it publicly available. The database consists of the

single file money.txt. Take a look at it: each line is a simple WAIS document of the form

*Country-name currency-name exchange-per-USdollar last-update*

Imagine, however, that the data is kept in an obscure data format that requires a custom parser, under the control of a customized DBMS that requires a filter program.

To integrate this "external" database with your WAISserver software , you first need to create a WAIS index of its contents. This can be done by using the example external database dump utility and the example custom parse format provided in the customcode directory.

With /wais/sample/currency-exchange as your current directory, create an index directory

```
% mkdir world-money
```

Now, go to the customcode directory

```
% cd customcode
```

and look around

```
% ls
ExchangeDump.exe      waisparse              currency-filter
currency-filter.wc          localParsers.c
```

ExchangeDump.exe is a c-shell script that dumps the contents of the money.txt database to standard output. View it in your editor. Notice that the dump script simply calls the UNIX cat command on the /wais /sample/currency-exchange/money.txt file. Double-check the pathname used by ExchangeDump.exe and change it if necessary; it will need to be changed if the default WAIS installation directory structure was not used.

Next, look through the copy of localParsers.c in the customcode directory. As you can see, after the sample functions, you will notice that it contains C functions and a defParser structure which together define a custom parser named currency-parse. This is a very simple parse format: it is just a copy of the standard one-line parse format, with two additions: 1) Gather and Finish Doc Key functions, and 2) code in the Document Separator function to collect the modification date. It is essential that doc-keys and modification dates be collected when parsing non-file based data. In the absence of filenames, doc-keys become the unique document identifiers. Similarly, without the date stamps of UNIX files, modification dates must be relied on for incremental indexing updates. The waisparse program in your

customcode directory was compiled with the custom currency-parse format included.

Before going on to examine the custom filter code, try using the dump program and customized parser to build an index for the currency exchange database.

From within the customcode directory, type the following command as a single, contiguous line:

```
% ExchangeDump.exe | waisparse -parse currency-parser +
        | waisindex -d ../world-money
```

Here, ExchangeDump.exe dumps the database contents to standard out, while the + option to waisparse allows it to read data from standard input. The customized waisparse in the customcode directory uses the currency-parse format and outputs the parsed data to standard output. The waisindex program accepts the data piped from standard out, indexes it, and creates WAIS index files in the directory /wais/sample/currency-exchange/world-money. You should see output similar to

```
10274: 0: Oct 7 16:26:01 1994: 100: indexed China
10274: 1: Oct 7 16:26:01 1994: 100: indexed Mexico
10274: 2: Oct 7 16:26:01 1994: 100: indexed Russia
10274: 3: Oct 7 16:26:01 1994: 100: indexed India
10274: 4: Oct 7 16:26:01 1994: 100: indexed Brazil
10274: 5: Oct 7 16:26:01 1994: 100: indexed Italy
10274: 6: Oct 7 16:26:01 1994: 100: indexed Thailand
10274: 7: Oct 7 16:26:01 1994: 100: indexed Poland
10274: 8: Oct 7 16:26:01 1994: 100: flushing 52 words (361 different words) to disk
10274: 9: Oct 7 16:26:01 1994: 6: building searchable dictionary
10274: 10: Oct 7 16:26:02 1994: 6: optimizing filename table
10274: 11: Oct 7 16:26:02 1994: 6: updating document and headline tables
10274: 12: Oct 7 16:26:02 1994: 6: committing changes
10274: 13: Oct 7 16:26:02 1994: 100: wrote source description ../world-money/world-
money.src, please examine it before use
10274: 14: Oct 7 16:26:02 1994: 100: building catalog
```

If not, recheck that you have typed the command correctly; it's easy to forget the + option to waisparse. Also, recheck that the DumpCurrency.exe script uses the correct pathname for money.txt.

Edit the world-money.src file and add text describing the currency exchange database. Also, check the other values and correct any that may be wrong, for example your email address.

Once you have successfully created an index for the currency exchange database, the next step in integrating it with WAIS is to tell the waisserver program about the custom filter program. Look now at the

file currency-filter. It is a simple c-shell script that abides by the filter program API, handling the arguments it receives from waisserver. In particular, it uses the doc-key to pick a document out of the database and send it to standard output.

To alert waisindex about this filter, you need to add to the index directory a configuration file that contains a FILTER declaration. Open the currency-filter.wc file. It contains one line:

**FILTER TEXT TEXT /wais/sample/currency-exchange/customcode/ currency-filter**

This is the filter declaration for the custom filter program currency-filter. It means that if a document retrieval request is recieved and the asked-for display format is TEXT and the available display format is TEXT, then waisserver calls the currency-filter program to retrieve the document. Before continuing, be sure to change the pathname in the filter declaration to match the currency-filter program's location in your directory structure.

The filter declaration belongs in a configuration file. With customcode as your current directory, type

**cp currency-filter.wc ../world-money/index.wc**

This creates a database-specific configuration file, index.wc, in the world-money index directory, where waisindex will find it when serving the currency exchange database. Notice that the filter declaration uses the full pathname to the filter program. When serving an external database it is absolutely critical to declare the full pathname because, if waisserver is running under inetd, there are no environment variables such as PATH, which tells UNIX what directories contain executables.

With the filter program thus installed, whenever a waisserver process searches the world-money index, it will use the currency-filter program to retrieve its documents.

Now you have completed the integration of the currency exchange database with WAIS. To test it, use the waislookup command. With currency-exchange as your current directory, type

% **waislookup -d world-money**@*your-wais-server-host*:*your-port*

You will see a response such as:

```
25094: 0: Oct  1 12:15:52 1994: 100: reading global configuration from
/usr/local/wais/wais.wc
local WAIS search on money-index
type: q      to quit
      r <N>  to retrieve document <N>
      f <N>  to feedback document <N>
      h <N>  to set maximum number of headlines to <N> (-1=all)
      anything else performs a search
>
```

At the > prompt, type your query, for example

```
> rupee
 0 score 1000 len    73 India rupee 31.1 08/24/94
 1 score   1 len   624 Query Report for this Search
> r 0
 India       rupee       31.1       08/24/94
> China
 0 score 1000 len    73 China yen 8.68 08/21/94
 1 score   1 len   620 Query Report for this Search
> r 0
 China       yen         8.68       08/21/94
> q
```

Here we show a query of rupee followed by a retrieval request for the one document that contains the word rupee. A second query and request sequence yeild information about the Chinese yen. Now we know that if one has thirty-one rupees, they can be exchanged for about eight and a half yen. Notice that queries for either the currency or the country name work equally well.

As the WAISserver administrator, you would update your external database index periodically. In the case of an international currency exchange database, the data is changing continuously. However, a bank sets its rates once a day. If this example were real, you would want to run an incremental indexing update daily — right after the bank had posted its daily rate. To do this, you would probably put a command such as

```
% CurrencyDump.exe | waisparse -parse currency-parser +
          | waisindex -d world-money -append
```

in a cron file and have it execute at the same time every day. The -append option to waisindex causes it to reindex only those documents that have changed since the last time waisindex was run on this database. If a country collapsed or was overtaken by another, you might need to run waisdelete to remove the corresponding currency exchange document from your index.

# WAISgate Connects WAIS with Web

The World-Wide Web (Web) is a popular Internet browsing and retrieval tool based on hypertext links between information sources. The WAIS Inc. WAISgate product is a gateway between WAIS and Web. If you are serving a WAIS database, WAISgate can greatly expand your user base by making your WAIS database(s) accessible from Web. Similarly, if you are serving a set of Web pages, WAISgate brings database searching capabilities to Web client users.

The World-Wide Web operates on a client-server model as does WAIS, but the two have different communication protocol and data format standards. WAISgate overcomes these differences. WAISgate receives Web requests and forwards them to your WAIS server, effectively creating a gateway between a Web Server and a WAIS server. WAISgate can handle requests from most Web clients that use Forms. This includes X-Mosaic, the most widely used Web client, and the character-based LYNX.

There are four major steps involved in installing WAISgate:

1. Install a WAIS server according to the instructions in this manual.
2. Install the NCSA Web Server, which can handle Forms-style requests from clients such as Mosaic that use the HyperText Transfer Protocol (HTTP). Complete Web Server documentation and installation instructions may be found by using X-Mosaic or LYNX to access the Web page

   **http://hoohoo.ncsa.uiuc.edu/docs/setup/Install.html**

   Note: the above address is called a Universal Resource Locator, or URL.
3. Install WAISgate in the Web executables directory, cgi-bin, as described in the file /wais/current/waisgate/INSTALL.
4. Translate your data source from WAIS to Web or vice versa:
   - If you have a WAIS database and are adding it to Webspace, then translate the WAIS Database source description file, *database*.src, into Hyper-Text Markup Language (HTML) by calling WAISgate with the -t option. Put the resulting *database*.html file in your Web hierarchy, have your Web Home page point to it, and create hypertext links to it from other appropriate places in your information store.

- If you have a hierarchy of Web pages and are adding WAIS search capabilities to it, then follow the instructions in Chapter 4, "Building a WAIS Database," to create a WAIS index for your Web data. When you invoke waisparse, use the html parse format. The parser will, by default, use the HTML display format. Now translate the *webdata*.src file into HTML format by calling WAISgate with the -t option. Put the resulting *webdata*.html file in your Web hierarchy, have your Web Home page point to it, and create hypertext links to it from other appropriate places in your information store.

In either case, the result will be Web pages that represent your data. As seen by Web client users, these will include query entry windows along with search and retrieval buttons. When a Web client user types in a query and clicks on a WAIS search button, the Web client process sends the WAIS query to your Web server. The Web server then calls WAISgate, which forwards the request to your WAIS server.

Complete instructions for installing WAISgate can be found on line in the file /wais/current/waisgate/INSTALL

# 7

# Monitoring WAIS Usage

## What Information is Logged

The WAIS server automatically records all transactions in the log file that you specified with the **-e** switch to **waisserver**. In our example WAIS server, the pathname of the file was set to /wais/logs/server.log. Generally, the information recorded in this file is so detailed that it is difficult to obtain a good measure of the usage patterns of your databases. Instead, usage characteristics are extracted from the log file and summarized using the WAIS Reporter as described in the next section. For the curious, however, the remainder of this section is devoted to a brief overview of the information recorded in the log file.

For each client process requesting service, the WAIS server records the WAIS server daemon process ID, the current count on the number of transactions performed for this client, the date, the time, and the type of transaction. The WAIS server records six main transaction types:

- Opening a connection
- Searching a database
- Returning results from a search
- Retrieving a document
- Closing a connection
- Errors and warnings

If a search transaction is performed, the server also records the name of the database and the client's question. If results were returned from a search, the number of documents found and the document identifiers are also logged. And finally, if a document is retrieved, the document identifier, the database name, the document size, and the document display format are all recorded.

# The WAIS Reporter

The WAIS Reporter can be used to summarize the information contained in the log file generated by the WAIS server. The generated summary, called the WAIS Usage Report, contains the following information:

- **Total number of connections**: This number represents the total number of independent client connections made to the WAIS server, where a connection equates to one **waisserver** process. A single connection can span over multiple searches and retrievals, and over multiple databases
- **Number of different machines connecting**: This number is the total number of client machines requesting services from this WAIS server.
- **Total number of searches**: This number is the total number of searches requested by all clients.
- **The total connect time (seconds)**: This number is the sum of the connect time of all clients. The connect time is the lifetime of each daemon server process, in seconds. The majority of the connect time is idle time.
- **The total search time (seconds)**: This number is the sum of the search time of each daemon server process, where the search time is the elapsed time, in seconds, that each daemon spends servicing its client's search request.
- **Searches returning zero hits**: This number is the total number of search requests resulting in no matches, where the server process was unable to find any documents matching the client's question.
- **Total number of documents retrieved**: This number is the total number of documents retrieved by all clients.
- **Total number of db's searched**: This number is the total number of different WAIS databases that clients have searched.
- **Number of searches with no DB name**: This number is the total number of times clients requested a search without specifying the database name. When a database name is not specified in the search request, the default INFO database is used.
- **Number of searches requesting help**: This number represents the total number of times a client process requested a search for "?" or "help". This gives you an idea of how many new users are

requesting information about the databases served on this machine.

- **Avg. number of seed words per search**: This number is the sum of the number of words contained in all questions divided by the number of questions, or search requests. The word count also includes boolean operators and stopwords.
- **Number of searches using rel feedback**: This number is the total number of searches performed with relevance feedback.
- **Number of server warnings**: This is the number of times a warning occurred while processing a client's request.
- **Number of server errors**: This is the number of times an error occurred while processing a client's request.

In addition to these totals, the WAIS Reporter also compiles lists of four more items:

- The total number of search and retrieval requests for each database searched by a client process. This information gives you a quantitative idea of the load on each database provided by the WAIS server.
- The names of all client machines accessing the server's databases and the number of connections requested by each machine.
- The names of the client software and the number of connections requested by clients using this software.
- The error and warning messages of any problems reported by the server.

# How to Use the WAIS Reporter

The WAIS Reporter program is invoked with the following command sequence:

**waisreporter** *log-file-name*

where *log-file-name* is the pathname of the log file. For example, in your invocation of the waisserver program, if you specified the **-e** switch with the filename /wais/logs, the call to the WAIS Reporter might look like:

```
% waisreporter /wais/logs/server.log
```

Normally the WAIS Reporter sends the summary report to the standard output. If you want to save the report out to a file, you can redirect the output to the file of your choice.

**waisreporter** *log-file-name* > *summary-file-name*

where *summary-file-name* is the pathname of the output file.

## Options to the WAIS Reporter

If your WAIS server services multiple databases, and you would like to see a WAIS Usage Report of a specific database, you can use the **-d** switch to specify the database you want summarized:

**waisreporter** *log-file-name* **-d** *database-name*

where *log-file-name* is the pathname of the log file and *database-name* is the pathname of the database's index files. The *database-name* is the same name specified in the argument list to **waisindex** and **waislookup**. A sample usage follows:

**waisreporter /wais/logs/server.log**
**-d /wais/indexes/resumes**

If the log file is stored as a compressed file, the - option is useful for piping the log information from the standard input. For example,

**zcat** *compressed-file-name*.**Z | waisreporter -**

This is particularly useful for analyzing archived log files.

## Sample WAIS Usage Report

An sample of a WAIS Usage Report is shown below.

```
WAIS Usage Report
generated on wais from /wais/logs/server.log-93-06-03
based on logs from Thu Jun  3 06:58:55 1993 to Thu Jun  3 18:42:26 1993

Total number of connections:            175
Number of different machines connecting:     2 (87.50 connections each)
Total number of searches:             113 (0.65 per connection)
Total connect time (seconds):         12741 (72.81 per connection)
Total search time (seconds):          391 (3.46 per search)
Searches returning zero hits:          5 (4.42%)
Total number of documents retrieved:        84 (0.74 per search)

Total number of db's searched:          5
Number of searches with no DB name:          3 (2.65%)
Number of searches requesting help:          4 (3.54%)
Avg. number of seed words per search:       0.11
Number of searches using rel feedback:       0 (0.00%)
Number of server warnings:              8
Number of server errors:                0

Database                                Searches  Retrievals
--------------------------------------------------------------------------------
catalogs                                    98          77
projects                                     9           7
INFO                                         3           0
resumes                                      2           0
weather                                      1           0
TOTAL                                      113          84

Client Machine                                      Connections
--------------------------------------------------------------------------------
saturn.wais.com                                         173
jupitor.wais.com                                          2
TOTAL                                                   175

Client Software                                     Connections
--------------------------------------------------------------------------------
                                                            8
WAIS-INC-Mac-WAIS1.1-GN                                     1
waissearch WAIS release 8 b2, from host: saturn.wais.com.             165
waissearch WAIS release 8 b5.1, from host: saturn.wais.com              9
TOTAL                                                   175

Error and Warning Report
--------------------------------------------------------------------------------
none
```

## Automating Reporting

A simple shell script can be used to manage the log file and to periodically invoke WAIS Reporter. An example of such a script is included with your installation in the file /wais/current/util/run-waisreporter. This section describes the operation of this shell script, and describes how to use the UNIX **cron** facility to periodically invoke the script.

The script operates as follows. First the server.log file in the /wais/logs directory is moved to a new file whose name is based on today's date,

(e.g. server.log-94-11-2.) Since the log file can become very large over time, segmenting it on a periodic basis helps to manage the size of the file. Next, an empty server.log file is created as a replacement, and the permissions of this file are changed to ensure read and write accessibility. And finally, **waisreporter** is run on the logs contained in the old server.log file, and the resulting WAIS Usage Report is mailed to the specified administrative personnel.

To set up a crontab entry that executes **run-waisreporter** on a periodic basis, first create a file, called *waisreporter-crontab-filename*, containing the following one-line entry[4]:

***minute hour* * * * *run-waisreporter-filename*** -e ***logfile*** -m ***mailing-list*** -b ***executable-directory***

where *minute* is the minute of the hour (0-59), *hour* is the hour of the day (0-23), and *run-waisreporter-filename* is the full pathname of the shell script. This format specifies that the **cron** program should be run on a daily basis. The **run-waisreporter** script takes three arguments, the location of the log file, *logfile*, the mailing list to send the query report, *mail-address*, and the directory location of the **waisreporter** program.

For example, the following command specifies a crontab entry that will execute the script in /wais/current/util/run-waisreporter daily at 11:55 pm.

**55 23 * * * /wais/current/util/run-waisreporter -e /wais/logs/server.log -m wais-admin@wais.com -b wais/current/bin**

To set up your crontab entry, as superuser, execute the following command:

**% crontab -e** *user*

where *user* is the name of a user account that will run the cron job. This will start an editor so you can enter the line above. To check that your crontab entry was set up properly, use the list option to crontab:

**% crontab -l**
**55 23 * * * /wais/current/util/run-waisreporter -e /wais/logs/server.log -m wais-admin@wais.com -b wais/current/bin**

To remove your crontab entry, use the **-r** switch to **crontab**.

---

[4]This example is based on a six field format. Check with your UNIX system administration manual to determine the exact format of your system's crontab.

# 8

# Clients and Queries

As a WAISserver administrator, you are primarily concerned with setting up and maintaining one or more WAIS databases. This chapter describes the other side of the WAIS equation: how a WAIS database is used.

First, the waislookup program is discussed; this is a very simple interactive client program. Next, we describe the kinds of queries clients can send to your WAIS server and how the WAIS search engine handles queries. Finally, the query report is discussed — this is the report your server returns to a client process along with each query to let the end user know how her search request was handled.

## Clients

WAIS Clients are user interface programs that enable end users to initiate search and retrieval requests for WAIS servers. The user of a WAIS client selects a database she wishes to search and then types a question or query. The client process sends the question to the WAIS server for the chosen database. This is called a search request. The server returns a results list consisting of relevance-ranked document headlines. Some clients also display the last modification date for each document. The end user may then request one of the listed documents and, if she does, the client sends a retrieval request to the server. The server responds by sending the document to the client, which displays it for the end user.

WAIS Inc. does not currently offer a WAIS client product, but freeware clients are available. The variations between client programs are mostly a matter of which platforms they run on and of how they look and feel. All available WAIS clients conform to a single network protocol suite and are all able to successfully communicate with the WAIS Inc. Server product. For a list of WAIS freeware clients, see Appendix D.

# Running Queries with waislookup

The waislookup program is included with the WAISserver software product. It operates in two distinct modes: as a local client and as a remote client.

## Using waislookup as a Remote Client

Using waislookup as a remote client, one can search any database listed in the public directory of servers. When invoked as a remote client, waislookup provides a simple user interface at which you may enter queries and retrieval requests and view results lists, query reports, and documents. This is a valuable tool for testing the operation of your server and getting a feeling for how various queries work against your database. Also, waislookup can provide end users at your site with a simple WAIS client, which may be all they need.

To invoke waislookup as a client, type

**waislookup -d** *database-name@server-host:server-port*

where *database-name* is the database name as published in the directory of servers (typically the name of the index directory for the database), *server-host* is the name of the machine on which the WAIS server is running, and *server-port* is the port number assigned to handle WAIS server connections. As you might suspect, this syntax works equally well for queries against remote databases as for those against your own server. For example

**waislookup -d directory-of-servers@wais.com:210**

will put you in a waislookup session ready to query the public directory of servers.

The waislookup program can use either Version 1 or Version 2 of the Z39.50 protocol. To try it out, simply set the environment variable connType to either Z39501988 (the default) if you're searching a freeware server or a WAIS Inc. WAIS server still at Version 1.0, and set it to Z3950V2 if you're searching a WAIS Inc. server at Version 2.0 or higher or a freeware server known to use the newer protocol.

You can use waislookup in combination with a WAIS Forwarder to search WAIS databases over firewalls. This is simply an extension to the client syntax for searching a remote database. For example, if WAIS Inc. had a firewall on the host machine firewall.yoyodyne .com, and

waisforwarder was running on port 210 there, you could search a database at the Directory-of-Servers at WAIS, Inc. by issuing the command:

**waislookup -d directory-of-servers@server.wais.com:210 \ @firewall.yoyodyne.com:210**

## Using waislookup as a Local Client

When invoked as a local client, waislookup bypasses the waisserver program and consults the index directory files directly, just as waisserver would. This is a valuable tool for testing a database before setting up the server or after maintenance tasks such as incremental indexing. To invoke waislookup as a server, simply type

**waislookup -d** *database-index-path*

where *database-index-path* is the full or relative pathname of the index directory for the database you want to search.

## The waislookup Interface

Whether you invoke waislookup as a local or remote client, you will see something like this:

```
25094: 0: Oct  1 12:15:52 1994: 100: reading global configuration from
/usr/local/wais/wais.wc
local WAIS search on world-money
type:  q    to quit
       r <N> to retrieve document <N>
       f <N> to feedback document <N>
       h <N> to set maximum number of headlines to <N> (-1 = all)
       anything else performs a search
>
```

At the > prompt, type your query, for example

```
> party AND fun NOT (T-shirts or Tee shirts)
```

You will receive a results list such as:

```
 0 score  260 len  2028 type TEXT 08/19/94: Susan Re: Party Wine
 1 score  496 len  2491 type TEXT 09/19/94: Brew Re: Party skit
 2 score  176 len  3565 type TEXT 08/25/94: davem Re: sw weenie
 3 score   96 len  1453 type TEXT 04/13/94: oveau Re:seamstress?
 5 score  106 len  2702 type TEXT 09/22/94: ma Re: Hey hey Moma!
 6 score   96 len  1370 type TEXT 09/19/94: ws Re: progress rpt
 7 score  127 len  1762 type TEXT 08/11/94: ord Re: Economist
>
```

Each document is given a number down the lefthand side; this is the number used to retrieve the document. Next in each row is the document's relevance score, preceeded by the word "score." The next section explains in detail how the WAIS search engine arrives at these

scores. Then comes the length, type, date, and finally the document's headline.

To retrieve one of the documents whose headline appears on the list, type r# at the > prompt, where # is the document's score. To use a document in relevance feedback, type f # at the > prompt. For a complete description of the waislookup interactive interface, see its command reference in Chapter 9.

# How Queries Work

At the heart of the WAISserver product is the WAIS search engine. The WAIS search engine receives a user's question or query, searches its database for documents most relevant to the question, and returns a relevance-ranked list of documents back to the user. Each document is given a score from 1 to 1000, based on how well it matched the user's question (how many words it contained, their importance in the document, etc.). A question, or query, is an expression that can contain any combination of natural language, literal strings, boolean operators, field name-value pairs, date or numeric ranges, and relevant documents.The WAIS search engine also supports right truncation (wildcard searching), word stemming, and relevance ranking. Each of these capabilities is explained in this section.

> Note: a copy of this section may be found on line in the file **/wais/current/**doc/howtosearch.txt.

## The Directory of Servers

The Directory of Servers is an index of all publicly usable WAIS databases. There are two copies one at The Center for Network Information Discovery and Retrieval (CNIDR) and one at WAIS, Inc. The Directory of Servers can be a very valuable part of your search strategy, because it's the only reliably up-to-date way for you to know what public WAIS databases exist.

Let's say you are interested in finding an organic pesticide for a species of beetle in Colombia. Where do you start? Usually, most people start by searching the Directory of Servers for "organic pesticide for a species of beetle in Colombia". Don't make this mistake! As easy as WAIS is to use, it still doesn't have all the answers. The Directory of Servers

contains the .src files for every public WAIS database. The .src file tells you where the database resides, which port, and who's the maintainer, and it has a short description of what kinds of documents are in the database. This short description won't ever have specific words like "beetle" or "pesticide" — rather, it will contain words like "agriculture", "insects", or "colombia". Many .src files have long lists of words that describe the categories of documents that are in the database.

Therefore, when you start out searching for an obscure topic, first try to find a WAIS database that deals with the broader category that the topic falls under. Then, when you've found a database that seems promising, search that database using the specific words; but even here, if the search turns up empty, try searching the directory for broad categories again to see if there are any general documents that would suggest where else to look -- for example, bibliographies, ftp sites, Mosaic pages, and so on. In short, try to think of other words to use, other ways to describe your information. If you strike out on your first few trys, don't assume WAIS can't help you find what you need.

## Natural Language

The server can be queried using natural language questions. The server does not understand the question, rather it takes the words and phrases in the question and finds documents that have those words and phrases in them. "Tell me about portable computers." is an example of a natural language question. In this example, the WAIS server would search for documents containing the words 'portable' and 'computers'; the other words, 'tell', 'me', and 'about', are called "stop words" -- words so common that they occur in almost every document and so they are not used for searching a document.

## Literal Strings

A similar but more specific kind of query asks to find documents that contain one or more exact phrases by enclosing them in double quotation marks. This is known as a literal. For example, the query

**"search engine capabilities"**

returns only documents that contain this exact phrase.

The WAIS search engine performs a literal search exactly as if you had used the boolean operator ADJ. Thus the above example would yield the same results as

**search ADJ engine ADJ capabilities**

For this reason, it is best to stick to noun phrases when using literals; if your literal phrase includes stopwords, the stopwords will be ignored.

## Relevance Feedback

Relevance feedback is the ability to select a document or a portion of a document and find a set of documents related to it. For example, suppose you perform a search on a news database with the natural language question "What's going on in personal computers?". Scanning the headlines returned, you see the headline, "Personal Computers in K-12", where you are interested in finding more articles related to this. You can then perform the search again using your original question, selecting this article, or a portion of the article, for relevance feedback. The search engine then returns a new list of headlines for related articles.

In essence, relevance feedback adds more words to the original question. These words are determined by finding the "significant" words in the document that was fed back to the server; the significant words are those that best distinguish it from all other documents. It then tries to find other documents that share these words.

One of the primary uses of relevance feedback is to help users quickly focus their search without the need to learn complex query languages. For example, you can use natural language to find a list of document headlines, and then use relevance feedback to focus your search on the documents most relevant. Since a WAIS search is fast, you can interactively and iteratively refine your search using a combination of natural language questions and relevance feedback.

## Boolean Operators

The boolean operators, AND, OR, NOT, and ADJ aid in establishing logical relationships between concepts expressed in natural language. These operators are especially useful in narrowing down the search.

**AND, &&**

The AND operator is helpful in restricting a search when a particular pair or larger group of terms is known. For instance, when searching for documents on the weather in Boston, a question such as "weather AND Boston" would return only those

documents that contain both the word "weather" and the word "Boston". You can use more than one AND in a query, e.g. "weather AND Boston AND November". Note that the C-like double ampersand (&&) may be used instead of spelling out the word AND.

**OR, ||**

The OR operator is often used to join two different phrases of a Boolean search. A question such as "hurricane OR tornado" would search for all documents containing either the word "hurricane", or the word "tornado", or both. You can also use more than one OR in a query. A natural language question is much like having an implicit OR between the words, except that the search engine does more work in a natural language query to determine the relevance of words and their relationships in a phrase.Note that the C-like double vertical bars (||) may be used instead of spelling out the word OR.

**NOT**

NOT is a binary operator. That is, it has to come between two or more words or parenthesized clauses. NOT is used to reject any documents that contain certain words. The question "basketball NOT college" would find all documents containing the word "basketball", that do not also contain the word "college". Note, however, that this question would eliminate articles on any professional players that mention their alma maters; in other words, be careful not to limit your search too much with the NOT operator, make sure that you know what you're throwing away.

**ADJ**

The adjacent operator, ADJ, is used to ensure that one word is followed by another in the returned document, with no other words in between. For example, "cordless ADJ telephone" returns only documents containing "cordless telephone" and ignores documents that only contain one of the words or that contain both but not adjacent to one another. ADJ will nonetheless work when stopwords interrupt two words; for example, the preceding example will find occurrences of "cordless for telephone". Note that the ADJ operator yields the same results as does a literal query. Also note that ADJ, unlike AND, OR, and NOT, is not a commutative property — "telephone ADJ cordless" does not work the same as "cordless ADJ telephone".

## Mixing Natural Language, Literals, And Booleans

The ability to mix natural language, literals, and boolean operators is unique to the WAISserver search engine. Combining natural language and boolean operators enables end users to better target their searches. For example, suppose you were looking for documents specifically on portable laptop computers that are not made by Tosuji Corporation. The question could then be "Tell me about portable laptop computers NOT Tosuji."

## Fielded Search

For data sets whose documents have special data fields, selected portions of the documents can be tagged by the WAIS parser as fields. A client can then ask a WAIS server to limit its search to those documents containing a user-specified value of a particular field. This is called a fielded search.

The mail-or-rmail parse format is an example of a parse format in which fields are tagged. For this parse format, the WAIS parser detects the "to" and "cc" fields, the "from" and "sender" fields, the "subject" field, and the "date" field. An example of a question using natural language, a boolean operator, and fielded search is: "company picnic AND from=barbara". The WAIS server would then find email messages about a company picnic that Barbara sent.

## Date and Numeric Ranges

For a date or numeric field, a range may be specified using the syntax

*field-name comparison-operator value*

where ***comparison-operator*** may be one of > (greater than), < (less than), <= (greater than or equal to), >= (less than or equal to), or = (equal to).

Currently, dates with the following formats are supported:

```
m-d-yy   m-d-yyyy   mm-dd-yy
m/d/yy   mm/dd/yy   m.d.yy
today    yesterday
```

and positive integers only are supported for numeric fields.

If the comparison operator is =, then the range may be specified using the word TO, as in

**date = 4/15/93 TO 4/14/94**

Both ends of the range are inclusively specified.

## Right Truncation (Wildcards)

A user can specify right truncation by ending a word with the asterisk (*) wild card character. This tells the search engine to search on words whose first several characters match the base characters before the *. For example, you might use right truncation in a question such as geo*, which may retrieve documents containing the words: geographer, geography, geologist, geometry, geometrical, etc.

## Grouping Search Terms

A user can group search terms and phrases together using parentheses.

For example, if you wish to search for information about snowstorms, tornadoes, or hurricanes in New York City, you might search for "(snowstorms OR tornadoes OR hurricanes) AND (New ADJ York ADJ City)." You can also nest your parentheses; for example, "from = ( (ben ADJ wais) OR (brewster ADJ think) )" searches for messages from either ben@wais.com or brewster@think.com. When you're using several boolean operators, you should always group, to disambiguate how the operators are to be applied.

## Relevance Ranking

Each document is scored based on its relevance to a user's question, where the most relevant document has the highest score, or rank -- 1000 being the highest, 1 being the lowest. A document receives a higher score if the words in the question are in the headline, if the words appear many times, or if phrases occur as they do in the question. A document's score is derived using techniques such as word weighting, term weighting, proximity relationships, and word density. These scoring techniques are outlined below.

## Word Weight

If a word in a document is found to match a word in the user's question, the word is assigned a weight, and this weight adds to the overall score of the document. The exact weight that a word receives depends on the emphasis given to the word by the author, and on where in the

document the word was found. For example, a word is weighted normally if it appears only in the text body with no capitalization, higher if the word has all capital letters or if the first letter of the word is capitalized, and highest if it appears in the headline. The WAIS parser determines word weights as it reads through the original data set.

## Term Weight

Each word used in a document is assigned a numerical value, called the term weight, based on the frequency of occurrence of that word over all documents in the data set. Words that occur frequently are not weighted as highly as those that appear less frequently. Very common words are either ignored or diminished in the scoring. For example, since the term, "animal", may occur frequently in many of the documents in a zoology data set, its term weight is small compared to a term such as "hippopotamus", which may occur only a few times.

## Proximity Relationships

Proximity relationship scoring specifies that if the words in a natural language question are located close together in a document, they are given a higher weight than those found further apart. The idea behind a proximity relationship is that if a document contains a phrase similar to one in the user's question, that document is more likely to be relevant.

## Word Density

The ratio of the number of times a word appears in a document to the size of the document is called the word density. It is a measure of how important a word is to the overall content of the document. A higher word density results in a higher relevance ranking for that document with respect to that word.

## Special Characters

The WAIS server was originally designed to be as general as possible and, in this spirit, it ignores all characters in a document that are not either an alphabetical letter or a number. In fact, non-alphanumeric characters usually separate words for the parser, for example, "F.Y.I." parses out to three words. This rule also applies to queries used to search a directory of servers.

### Stemming

Stemming is a technique used to automatically derive variations of a queried word. These variations are then used as part of the search. If stemming is used, then when a data set is indexed, word stems are indexed where possible. For example, "dancing," "danced," and "dancer" would all be indexed as "dance." A question containing the word "dancer", would then turn up documents that may also include "dancing", "danced", and "dancing".

Two types of stemming are supported: Plural and Porter stemming. Plural stemming attempts to determine the singular form of a word. Porter stemming attempts to find the real base, or stem, of a word and derive any possible alternate variations.

Since WAIS Inc. servers allow either form of stemming for a given database, be sure to ask the administrator of a database whether stemming is used before you search it. For example, you may search for "lens*" when curious about telescopes, but a plural stemmer will reduce your query to "len*" and return undesirable hits. A worst-case scenario is when you search on "s*" in a plural-stemming database; you will find no results, because the server stemmed your query to "*", the empty string.

## Query Report

A query report is a document created by the waisserver program that describes how an end user's question is parsed by the server. When a client process sends a query, the server creates and returns a query report to the client. The query report is the last document in the relevance-ranked list of documents returned by the server. The headline of the query report is listed as 'Query Report for this Search', and its relevance score is 1. Since the query report is an actual document, it may be retrieved for viewing by the client.

The query report contains the following information:

- The database being questioned
- The original question
- The boolean equivalent of the question in infix notation. This notation is a fully-parenthesized version of the question, showing the boolean operator precedence.

- The boolean equivalent of the question displayed as a tree
- The number of documents and the number of words in the database
- The number of unique words in the database
- The number of times each word in the question occurred in the database
- The expanded search words resulting from right truncation
- The number of documents found that satisfied the question
- The amount of elapsed time it took to perform the search

The purpose of this information is to give the user feedback on how the question was interpreted by the server, and on how well the information in the database matched the words in the question.

Below is an example query report generated from the following question: "carbon monox* AND poison". Simply stated, the question is looking for documents concerning both carbon monox* and poison*. The question uses right truncation for monox* and poison* to match words such as monoxide, monoximes, etc., and poisoned, poisoning, and poisonous.

```
Headline: Query Report for this Search
This is the search report for the search you ran on Jun  3 13:31:51 1993.
It is a temporary file, and will expire about an hour after the search.

----------------------------------------------
Searching /proj/wais/wais-sources/doe...

Your query:

 carbon monox* AND poison*

is equivalent to:

 ((carbon monox*) AND (poison*))

and was interpreted as:

 AND
  ( carbon monox*
    poison*
  )

The database contains 39,062,401 words in 230,750 documents.
There are 639,200 different words.

carbon occurs 30,404 times in 14,896 documents.
monox* is expanded to:
  monoxide occurs 3,825 times in 2,515 documents.
  monoximes occurs 1 time in 1 document.
  monoxodithioacetal occurs 1 time in 1 document.
  monoxygenases occurs 2 times in 2 documents.
  monoxyhemoglobin occurs 1 time in 1 document.
poison* is expanded to:
  poisoned occurs 61 times in 49 documents.
  poisoning occurs 486 times in 283 documents.
  poisonous occurs 17 times in 14 documents.

The search found 67 documents.  It took about 5 seconds.

----------------------------------------------

The search was performed by a WAIS Inc server:  WAIS waisserver 1.0.9i.
For more information email info@wais.com.
```

The query reporter uses the *db-index*/wais-tmp directory for temporary storage of the query report, where *db-index* is the directory specified in the -d switch to waisserver. Query reports are automatically deleted after about 1 hour.

# 9

# Command Reference

The following sections are the UNIX manual pages describing the WAIS programs. They are available online in the /wais/current/man directory. To read one of these manual pages online, type

% **man** *command-name*

where *command-name* is the name of the WAIS command you want to know about.

## waisdelete

NAME

waisdelete - mark documents for removal from a database.

SYNOPSIS

waisdelete
-d *index_filename*
[ -x *docKey* [*start-byte length-in-bytes*] ]
[ - reads arguments from stdin ]

DESCRIPTION

waisdelete marks documents for deletion from the wais index. Documents so marked are not returned as part of a search, even if they would otherwise match the search criterion. The resources used to record them in the index file are not recoverd until new documents are added to the database using waisindex -append or -merge. In no case are the original documents modified or removed.

OPTIONS

**-d** *databasename*
The **-d** switch specifies the pathname of the index files.

**-x** *docKey* [*start-byte length-in-bytes*] ]
Specifies a document to delete. If the index was created from files in the file system, docKey is the full path name to the file to mark as deleted. If the file contains several documents (eg. a mail archive file), start-byte and length-in-bytes are used to identify which document in the file to delete. If the document was built from an external database, the *docKey* is the *docKey* assigned

by the custom parser.

- Directs **waisdelete** to read arguments from stdin. Each argument must be on a separate line.

DIAGNOSTICS

Exit status 0 if successful, 1 if errors are encountered.

BUGS

There is no way to flush deleted documents from an index, other than using **waisindex** to **-append** or **-merge** new documents.

There is no waisundelete.

SEE ALSO

**waisindex, waislookup, waisparse, waisreporter, waisserver**

# waisindex

NAME

**waisindex** - index a stream of documents for fast search and retrieval

SYNOPSIS

**waisindex**
**-d** *index_filename*
[ **-mem** *megabytes* ]
[ **-sw** *filename* ]
[ **-stem** *stemmer-name* ]
[ **-nosrc** ]
[ **-nocat** ]
[ **-append** ]
[ **-v** ]

DESCRIPTION

**waisindex** takes a stream of documents from stdin, typically generated by waisparse, and creates an index for fast search and retrieval of those documents. The final index requires approximately 1/3 to 1/2 of the memory space of the original data. In addition temporary space equal to twice that is required during indexing.

OPTIONS

**-d** *database-name*

The **-d** switch specifies the pathname of the index files. For example, if **/wais/indexes/foo** is specified for *database-name*, then the index files will be located in the directory **/wais/indexes/foo** and named **/wais/data/indexes/foo/index.***. If the directory does not exist, **waisindex** creates it. If the directory exists and already contains index files, the index files are overwritten. For indexing speed, *database-name* should be a directory on the local file system of the machine running **waisindex**. Indexing works over NFS, but it is significantly slower.

**-mem** *megabytes*

The **-mem** switch specifies the number of megabytes of main memory the indexer can expect to use for indexing. The more main memory available to the indexer, the faster the indexing occurs. The default value is 10. For example, if the indexing machine has 64 or 256 Mbytes of main memory, it may be reasonable to use a value of 50 or 200, respectively. It is very important that the size you choose is available to waisindex in RAM. Otherwise waisindex will performance will be severely hurt.

**-sw** *filename*

The **-sw** switch specifies a file that contains a list of words to be used as stopwords, where each line of the file contains one stopword. Stopwords

are ignored in the creation of the index. During natural language and relevance feedback search, the stopwords have no effect. During boolean search, they are treated as if they have never occurred. The default stopword list is located in **/wais/current/util/stopwords.txt**, and may be modified as necessary. **/dev/null** may be used to specify *no* stopwords.

**-stem** *stemmer-name*
The **-stem** switch specifies the stemming algorithm to use on the database. Stemming is a technique used to automatically derive variations of a queried word. These variations are then used as part of the search. If a question contains the word "skate", for example, stemming is used to find documents that may also include "skates", "skated", and "skating". *stemmer-name* can be either plural or porter. Plural stemming searches for both the word and its corresponding plural. Porter stemming attempts to find the real base, or stem, of a word and derive any possible alternate variations. The default is no stemming.

**-nosrc** Inhibits the creation of a source file. If a source file already exists, the file is not overwritten, regardless of whether or not this option is given.

**-nocat** Inhibits the creation of a catalog. This option is useful for databases with a large number of documents, since the catalog contains several lines per document and may become large. A catalog is most useful for small databases. When a search returns no relevant documents, the catalog is returned.

**-append** The **-append** switch tells the indexer to append to the database specified by *database-name*, instead of overwriting it. If the indexer encounters a document has not been previously indexed, it appends the document to the index. If a document is from a file that has been modified since the last indexing, the old document is deleted, and the newer version of the document is appended to the index.

**-unlock** Unlocks a database by removing update and write locks. This is useful for removing locks left behind when an indexing job dies or is aborted.

**-nice** *level*
Runs the indexer at the given priority level. Level can range from -20 to 20 where higher numbers are lower cpu priority (i.e., nicer). The default value is 0. Only the superuser can use negative levels.

**-v** Print the version of the waisindex program.

ADVANCED FEATURES

**-help** Prints additional usage information.

**-lock** *mode*

Locks the database according to the *mode*. The *mode* is either **read**, **update**, or **write**. Write indicates that the index files will be changing immediately. It can only be placed when there are no other locks. Update indicates that the database is not to be modified, by other processes, and that you intend to place a write lock in the future. It allows read locks to be placed, but not write or locks. Read indicates that the files are being read and should not be modified. If the requested lock is not available the program will wait until it becomes available. The locks exist for the duration of the process only.

**-merge** *db1 db2*

Merge *db1* and *db2*, creating the database given in the **-d** argument. *Db2* must not contain any deleted documents. The result is built as per the **-tmp** flag. Use **-finalize** to make it searchable.

**-replacewith** *new-db*

Replace the db given in in the **-d** argument with the *new-db*. The old db is destroyed. This command uses the locking protocol to ensure that users are not disrupted during the replacement.

**-tmp** Build the database without a searchable dictionary, and without optimization, and don't generate .cat or .src files. This saves index time and is useful when creating or merging several temporary files.

**-finalize** Create a searchable dictionary and optimize the index of the database given in the -d argument. Also create the .src and .cat files unless -nosrc and -nocat are specified.

DIAGNOSTICS

Exit status 0 if successful, 1 if errors are encountered.

BUGS

The second argument to **-merge** must not contain any deleted documents.

Solaris 2.3 requires patches to non Solaris 2.3 NFS servers in order for locking to work correctly over NFS.

Before using **-append**, the database must have been created using some other option to waisindex.

SEE ALSO

**waislookup, waisparse, waisreporter, waisserver, waisdelete**

# waislookup

NAME

**waislookup** - an interactive interpreter for searching and retrieving documents directly from a local WAIS database, or from a server via the network.

SYNOPSIS

**waislookup**
**-d** *database-name*
[ **-config** *configuration-file* ]
[ **-v** ] **waislookup**
**-d** *database-name@server-host:server-port*
[ **-config** *configuration-file* ]
[ **-v** ]

DESCRIPTION

**waislookup** is a simple WAIS client which can be used to query databases in two ways. In the first form, where only the database name is specified, **waislookup** searches the named db directly, using the same search engine used by **waisserver**. This is useful for testing new databases. If the database name references a server (using the @host:port syntax), a z39.50 connection is made to that server, and the search and retrieval are performed through it. In this way **waislookup** can be used to test local or remote servers.

OPTIONS

**-d** *database-name*

Directs **waislookup** to search *database-name,* which is the pathname of a database on the local machine. It is the same name as specified in the **-d** switch to **waisindex**. The search is performed directly by **waislookup**, without going through a **waisserver**

**-d** *database-name@server-host:serverport*

Directs **waislookup** to search the database named *database-name* by connecting to a waisserver running on *server-port* port of the machine named *server-host*. All search and retrieval requests are passed to the server using the network connection, and the work is done there. The *:server-port* may be omitted, in which case it defaults to port 210. Normally the connection is made using the z3950-1988 protocol. If the remote server supports the WAIS Profile of the z3950-V2 protocol, you can use it by setting the **connType** environment variable to Z3950V2.

**-config** *configuration-file*

Read the global configuration from configuration-

file. By default the global configuration is read from **wais.wc** (in the current directory), **/usr/local/wais/wais.wc**, **/usr/local/wais/etc/wais.wc**,or **/etc/wais.wc**, whichever is found first.

**-v** Print the version of the waislookup program.

USAGE

The **waislookup** interpreter provides five interactive commands: retrieve, feedback, headline, and quit. By default, a question typed at the prompt followed by a carriage return is interpreted as a search command The question is either a natural language query, a boolean expression, or combination thereof. **waislookup** responds by returning a relevance-ranked list of documents from the database. Each document in the list is consecutively numbered with the document numbered 0 as the most relevant document.

After performing a search, the text of any of the listed documents can be retrieved by specifying the number of the document to be retrieved. Specifically, to retrieve a document numbered *n*, type an **r** character followed by a blank space, the integer value *n*, and a carriage return. Some documents may be available in multiple formats, in which case the format numbered *m* of the document numbered *n* can be retrieved by specifying *r n.m*. For example, to retrieve the second format of the third document, issue the commmand **r 2.1** at the **waislookup** prompt.

Relevance feedback is performed in the same way as retrieval, except **f** is used instead of **r**. The most important words of the specified document are then used to find additional documents that are "similar" to the original document.

To limit the number of headlines returned by a search to *n*, at the command prompt type an **h** character followed by a blank space, the integer value *n*, and a carriage return. A value of **-1** specifies that all documents should be returned. The default value is 40.

To quit the **waislookup** interpreter, type a **q** followed by a carriage return.

SEE ALSO

**waisindex, waisparse, waisreporter, waisserver, waisdelete**

# waisparse

NAME

**waisparse** - parse data into a stream of documents for use by **waisindex**

SYNOPSIS

**waisparse**
[ **-parse** *parse-format* ]
[ **-c** ]
[ **-l** ]
[ **-display** *display-format* ]
[ **-contents** **-nocontents** ]
[ **-assoc** *extension display-format* ]
[ **-wl** *length* ]
[ **-nf** ]
[ **-ph** ]
[ **-v** ]
[ **-r** ]
[ **-** ]
[ **+** ] *files-or-directories-to-parse*

DESCRIPTION

**waisparse** reads a set of data from files, directories, or stdin, and separates it into a stream of documents. For each document, the parser identifies the headline, field information, words, and docid. These are written to stdout, which is typically piped into waisindex to complete the building of a WAIS database. The original data is not modified.

OPTIONS

**-parse** *parse-format*
The **-parse** switch tells the parser how to extract the headline, field information, words and docid from the input data. The default parse format is text, in which a whole file is the document, and the headline is the filename. To display a list of the parse formats supported by WAIS Inc, run waisparse with no arguments.
Each time a new **-parse** format is specified on the command line, any switches set up to that point are reset to their default value. This allows several different parse formats to be specified in a single call to waisparse.

**-c** The **-c** switch prints a list of the parse formats contributed by the freeware community.

**-l** The **-l** switch prints a list of the parse formats developed locally at your site.

**-display** *display-format*
**waisparse** associates a display format with each document. The display format is ultimately used

by the WAIS client user interface to determine how to display the document. When a parse format is not specified, the default display format is TEXT. Otherwise, the default display format is determined by the parse format. For example, if the parse format is gif, then the default display format is GIF.

**-contents (-nocontents)**
Include (exclude) the contents of the file from the index. The filename and headline will always be included. The default is dependent on the parse format specified.

**-assoc** *extension display-format*
The **-assoc** switch specifies the extension and display format of secondary files that should be associated with the files-to-parse. Secondary files are not parsed, but are returned as part of any search which matches the files they are associated with. For example, this switch can be used to associate an image file with a descriptive text file. Only the words in the text file are parsed, but when a client performs a retrieval, the server returns both the text file and the image file. *Extension* is the filename extension of the secondary files, and *display-format* is the display format of the secondary files.

**-wl** *length*
The **-wl** switch specifies the minimum length (in characters) of words identified by **waisparse**. Words having fewer characters are ignored by the parser. The default length value is 1.

**-nf** Ignore fields in the data. This option is useful if the selected parse format identifies fields and if it is necessary to reduce the storage requirements of the indexed files. The storage of field information increases index file size by roughly 25%.

**-ph** Print headlines of documents as the parser runs. This is useful for debugging new parsers.

**-v** Print the version of the **waisparse** program.

**-r** Recursively parse subdirectories. Note: names give in *files-or-directories-to-parse* must contain directory names for this option to work properly. For example directories **A** and **B**, containing files **A.txt** and **B.txt** can be parsed using waisparse **-r A B**, but not waisparse **-r *.txt**, since ***.txt** does not match either **A** or **B**. If more complex operations are necessary, use the UNIX **find** command to generate filenames to pipe into **waisparse** using the **-** option (see below).

**-** Take the names of the files to parse from standard input, one file per line.

**+** Read the data from standard input. This is typically used when parsing data from external data-

bases through a database dumping program.

*files-or-directories-to-parse*

This is a list of the pathnames of files or directories to be parsed by *waisparse*. If the WAIS database is to be used from a machine other than the machine on which the parsing and indexing is performed, the filenames should be specified with a machine independent pathname.

EXAMPLES

To parse and index a database named manuals, consisting of a set of files, where the filenames are the headlines, and all the words are indexed. Execute the following:

```
% waisparse /product/manuals/*.doc.[ch] |
      waisindex -d /wais/indexes/manuals
```

To index a UNIX mail file containing many messages, where each message is a separately document, and the headline is the subject field of the mail message, execute the following:

```
% waisparse -parse mail-or-rmail /user/elmer/mail |
      waisindex -d /user/elmer/indexes/mail
```

To parse and index a set of files with the same prefix, descending down subdirectories:

```
% find /usr/http/htdocs -name '*.html' | waisparse -parse
html - | waisindex -d /usr/wais/html-docs
```

To index Microsoft Word documents so they can be displayed on a PC or a Macintosh, the parse format is text and the display format is MS-WORD. Text format a the parse format where the filename is the headline, and the words of the file are put into the index. Although there may be special formatting instructions in the document, these extra words do not amount to much space in the index, and do not effect search and retrieval. The display format of the documents is MS-WORD so that the client will know how to display it. The command is:

```
% waisparse -parse text -display MS-WORD /user/elmer/*.doc |
  waisindex -d /user/elmer/indexes/ms-doc
```

When indexing images where there are no words in the file, only the filename can be used to provide terms for the index. Use the filename parse format:

```
% waisparse -parse filename -display GIF /user/elmer/*.gif |
  waisindex -d /user/elmer/indexes/weathermaps
```

Since the display format is GIF, a parse format of gif could have also been specified. In this case, the gif parse format automatically sets the display format to GIF.

```
% waisparse -parse gif /user/elmer/*.gif |
  waisindex -d /user/elmer/indexes/weathermaps
```

An example of associating a set of text files with a set of gif and tiff image files is:

```
% waisparse -assoc gif GIF -assoc tiff TIFF /user/elmer/*.txt |
  waisindex -d /user/elmer/indexes/weathermaps
```

This associates files matching ***.txt** (**/usr/elmer/foo.txt**, for example) with gif image files with the same name but the **.gif** extension (**/usr/elmer/foo.gif**) using the display format GIF, and with files with the tiff extension (**/usr/elmer/foo.tiff**) and the TIFF display type. The headline and words are taken from the .txt file. In this case, since the parse format is text, the headline would be "foo.txt".
Files with different parse types can be specified in the command line. For example:

```
% waisparse -parse text -display MS-WORD /papers/*.doc
        -parse ps /press-materials/reports.ps
        -parse filename -display MIME /slides/*.sl
        | waisindex -d /wais/indexes/company-info
```

within the same set of data. It also shows that the **-parse** switch resets any subsequent switches to their default value. Specifying **-parse ps**, for example, reset the display format to the default, which is PostScript in this case.

BUGS

The **-r** option is limited. Use the UNIX find command for complex cases.

SEE ALSO

**waisindex, waislookup, waisreporter, waisserver, waisdelete**

# waisreporter

NAME

**waisreporter** - generate a WAIS Usage Report

SYNOPSIS

**waisreporter**
[ **-d** *database-name* ]
[ **-sr** ]
[ **-ne** ]
[ **-nw** ]
[ **-pul** ]
[ **-cli** ]
[ **-v** ]
*log-file-name*

DESCRIPTION

**waisreporter** summarizes the contents of a log file generated by a WAIS server. The summary lists the number of searches and retrievals performed on each database, the total number of connections, the number of different machines connecting, the total connect time, the total search time, etc. It also lists any errors or warnings encountered by the WAIS server.

OPTIONS

**-d** *database-name*
The **-d** switch specifies the name of the database that is to be summarized. If this argument is not specified, a summary of all databases recorded in the log file are generated.

**-sr** Print a detailed report on the searches performed.

**-ne** Suppresses reporting of any error messages from the log file.

**-nw** Suppresses reporting of any warning messages from the log file.

**-pul** Print any entries log entries which **waisreporter** can not parse.

**-cli** Clean up the client identification strings. The name of a client software program is sent to the server in an init message. Some clients also include IP number or date information, which is recognized and stripped by this option.

**-v** Print the version of the **waisreporter** program.

*log-file-name*
This is the pathname of the log file that waisreporter is to summarize. If a '-' is specified, the log file is taken from the stdin.

# waisserver

NAME

**waisserver** - service WAIS search and retrieval requests

SYNOPSIS

**waisserver**
[ **-p** *port* ]
[ **-e** *log-file* ]
[ **-t** *seconds* ]
[ **-u** *user* ]
[ **-args** *file* ]
[ **-v** ]
[ **-d** *directory* ]
[ **-config** *configuration-file* ]

DESCRIPTION

**waisserver** is a program that services WAIS clients using the Z3950 and TCP protocols. For each new client that contacts the server, the server forks off a child server process to handle that client's requests. The child server remains active until the client closes the connection. If the server is started from the command line or from a shell script, the server is in standalone mode.
If the name of the process is **waisserver.d** then it is assumed to have been started from the **inetd** daemon. This mode of server operation is called daemon mode. See the examples below for how to set up the **/etc/inetd.conf** and **/etc/services** files.

OPTIONS

**-p** *port* The **-p** switch specifies the port number at which the server waits for client connections. The default port number is 210. If the port number specified is less than 1024 (including 210), then the user must be root.

**-e** *log-file*
The **-e** switch specifies the name of the log file where the server records all transactions. If the server is run in standalone mode, the default is to send the log messages to stderr. If the server is run in daemon mode, the default is to send the log messages to **/server.log**.

**-t** *seconds*
Specifies the number of seconds to wait before timing out an idle client. The default is 36000 seconds (10 hours).

**-u** *user* The **-u** switch specifies the user account which will run the server. You must be root to use the **-u** option. If no user is specified, the server runs as the user who started it.

**-args** *filename*
The **-args** specifies a file from which arguments will be read. The file is made up of valid waisserver arguments, one per line. When the end

of the file is reached, it is closed, and the server continues reading arguments from the command line. This switch is required in order to pass more than 5 arguments to a server running under inetd (in daemon mode).

**-v** Print the version of the waisserver program.

**-d** *directory*
The **-d** switch specifies the base directory in which the default databases served by this server exist. If directory is not specified, and if the server is run in standalone mode, the directory defaults to the current working directory. If directory is not specified, and the server is run in daemon mode using the inetd daemon, directory defaults to /.

**-config** *configuration-file*
Read the global configuration from *configuration-file*. By default the global configuration is read from **wais.wc** (in the current directory), **/usr/local/wais/wais.wc**, **/usr/local/wais/etc/wais.wc**, or **/etc/wais.wc**, whichever is found first.

EXAMPLES

In standalone mode, the following command,

**% waisserver -p 8000 -d /wais/indexes**
**-e /wais/logs/server.log**

runs the server on port 8000, with the default databases directory **/wais/indexes**, logging messages in the file **/wais/logs/server.log**. For a WAIS client to communicate with this server, its port must also be set to 8000. Since the port number is greater than 1024, the waisserver command could be executed by any user.
The equivalent command using **-args** is:

**% waisserver -args /wais/server.args**

where **/wais/server.args** contains the lines:

**-p 8000 -d /wais/indexes -e /wais/logs/server.log**

To run the server in daemon mode, an example entry in the **/etc/inetd.conf** file is:

**z3950 stream tcp nowait root /wais/current/waisserver**
**waisserver.d -d /wais/indexes -e /wais/logs/server.log**

**Note:** the **inetd** entry must be on a single line.

The corresponding entry in **/etc/services** is:

**z3950 210/tcp # Wide Area Information Server (WAIS)**

To activate this new server, send a **HUP** signal to the **inetd** process.

**NOTE**: the **inetd** configuration specifies the user and port so there is no need to user the **-u** or **-p** switches when running under **inetd**.

DIAGNOSTICS

Exit status 0 if successful, 1 if errors are encountered.

SEE ALSO

**inetd, inetd.conf, waisindex, waislookup, waisparse, waisreporter, waisdelete**

# Appendices

# A

# WAIS Quick Start

So you are a UNIX wizard and have decided to jump right in and install the WAISserver software. Great! This section will show you what is needed and give a real life example. We'll keep the commentary to a minimum so you can bring your server up as soon as possible.

Note that in our examples, the current WAIS release is wais-2-0. The actual release which you are unpacking might have a different name, so whenever you are asked to type wais-2-0, replace it with the name of the version you are using.

## Installation

Let's get started. The first step is installing the WAIS software onto your UNIX machine. You'll need about 5 megabytes of free space to install the software. For the examples in this section, we'll use /wais as our root directory. Once you've decided where to install the software, make the directory and move into it.

```
% mkdir /wais
% cd /wais
```

If you've **ftp**'d the software over the Internet, you'll need to uncompress it and unpack it. The file is compressed if its filename ends in .Z, and it is an archive if it has a .tar extension. Here's the line that does the work:

```
% mv wais-2-0.tar.Z /wais
% cd /wais
% zcat wais-2-0.tar.Z | tar -xvf -
```

If the software is on a tape, you'll have to copy the tape onto your hard disk. That means you will need to know which UNIX device describes the tape drive on your system. Your system administrator should be able to help with this. We'll use a typical one in this example; it is the first tape drive on the system and it's called /dev/rst0 on SunOS 4.x, and /dev/rdsk/c0t4d0s2 on SunOS 5.1

```
% cd /wais
% tar -xvf /dev/rst0
```

In either case, the computer should print something like this:

```
x wais-2-0/bin/waisparse, 270336 bytes, 528 tape blocks
x wais-2-0/bin/waisindex, 720896 bytes, 1408 tape blocks
x wais-2-0/bin/waisdelete, 712704 bytes, 1392 tape blocks
x wais-2-0/bin/waisserver, 1474560 bytes, 2880 tape blocks
x wais-2-0/bin/waislookup, 1269760 bytes, 2480 tape blocks
x wais-2-0/bin/waisreporter, 172032 bytes, 336 tape blocks
x wais-2-0/bin/README, 127 bytes, 1 tape blocks
x wais-2-0/util/update-lcl-dir-of-servers, 1024 bytes, 2 tape blocks
x wais-2-0/util/run-waisreporter, 1947 bytes, 4 tape blocks
x wais-2-0/util/stopwords.txt, 2002 bytes, 4 tape blocks
x wais-2-0/util/filter-codes.h, 1544 bytes, 4 tape blocks
x wais-2-0/util/README, 218 bytes, 1 tape blocks
x wais-2-0/util/how-to-search, 15368 bytes, 31 tape blocks
x wais-2-0/doc/waisparse.txt, 10432 bytes, 21 tape blocks
x wais-2-0/doc/waisindex.txt, 8750 bytes, 18 tape blocks
x wais-2-0/doc/waisdelete.txt, 2243 bytes, 5 tape blocks
x wais-2-0/doc/waisserver.txt, 5697 bytes, 12 tape blocks
x wais-2-0/doc/waislookup.txt, 5580 bytes, 11 tape blocks
x wais-2-0/doc/waisreporter.txt, 2281 bytes, 5 tape blocks
x wais-2-0/doc/access-lists.txt, 1726 bytes, 4 tape blocks
x wais-2-0/doc/ethics.txt, 9338 bytes, 19 tape blocks
x wais-2-0/doc/Release-Notes-prelim.msword, 27648 bytes, 54 tape blocks
x wais-2-0/doc/Admin-Manual-2.0-prelim.msword, 318464 bytes, 622 tape blocks
x wais-2-0/man/waisparse.1, 8180 bytes, 16 tape blocks
x wais-2-0/man/waisindex.1, 6475 bytes, 13 tape blocks
x wais-2-0/man/waisdelete.1, 1679 bytes, 4 tape blocks
x wais-2-0/man/waisserver.1, 4414 bytes, 9 tape blocks
x wais-2-0/man/waislookup.1, 4004 bytes, 8 tape blocks
x wais-2-0/man/waisreporter.1, 1700 bytes, 4 tape blocks
x wais-2-0/sample/README, 3335 bytes, 7 tape blocks
x wais-2-0/sample/data/resumes/res3.txt, 4102 bytes, 9 tape blocks
x wais-2-0/sample/data/resumes/README, 5243 bytes, 11 tape blocks
x wais-2-0/sample/data/resumes/res1.txt, 3070 bytes, 6 tape blocks
x wais-2-0/sample/data/resumes/res2.txt, 4634 bytes, 10 tape blocks
x wais-2-0/sample/data/mail/rmail, 5760 bytes, 12 tape blocks
x wais-2-0/sample/data/mail/README, 4394 bytes, 9 tape blocks
x wais-2-0/sample/data/jargon/README, 4292 bytes, 9 tape blocks
x wais-2-0/sample/data/jargon/convert.c, 1874 bytes, 4 tape blocks
x wais-2-0/sample/data/jargon/jargon.231.text, 558266 bytes, 1091 tape blocks
x wais-2-0/sample/data/weather/ACUS1.DOC, 4961 bytes, 10 tape blocks
x wais-2-0/sample/data/weather/README, 2866 bytes, 6 tape blocks
x wais-2-0/sample/data/weather/WXKEY.DOC, 31 bytes, 1 tape blocks
x wais-2-0/sample/data/weather/WXMAPARS.DOC, 36 bytes, 1 tape blocks
x wais-2-0/sample/data/weather/ACUS1.GIF, 13581 bytes, 27 tape blocks
x wais-2-0/sample/data/weather/SELSLOG.GIF, 13870 bytes, 28 tape blocks
x wais-2-0/sample/data/weather/WXKEY.GIF, 18865 bytes, 37 tape blocks
x wais-2-0/sample/data/weather/WXMAPARS.GIF, 35661 bytes, 70 tape blocks
x wais-2-0/sample/data/weather/SELSLOG.DOC, 5216 bytes, 11 tape blocks
x wais-2-0/sample/Makefile, 1327 bytes, 3 tape blocks
x wais-2-0/sample/currency-exchange/customcode/waisparse, 253952 bytes, 496 tape
blocks

...and so on.
```

When you set up your WAIS server and WAIS databases, you won't want to encode the version number in each configuration file, since that would mean reconfiguring each file when a new release is installed. Instead we recommend making a symbolic link which maps the name of the release directory into a standard name used by the configuration files. In our example, we'll use "current" as the standard name.

```
% ln -s wais-2-0 current
```

When a new release is installed (e.g. wais-2-0-1), you can switch over to the new software by removing the old link, and making a new one.

To install the WAIS programs and UNIX manual pages, add the following to your .cshrc file:

```
setenv PATH /wais/current/bin:$PATH
setenv MANPATH /wais/current/man:$MANPATH
```

and type the following at the command line

```
% source ~/.cshrc
```

The installation applies to C shell (csh) users only. If you use another shell (e.g. borne shell, korn shell, etc.), you will need to modify the installation to suit your shell.

Now let's take a look at the software just installed. The **ls** command does a directory listing.

```
% ls current
bin             doc             parser-toolkit  util
client-toolkit  man             sample          waisgate
```

The bin directory contains the WAIS executables., the doc directory contains documentation, the parser-toolkit directory contains the source code and documentation for the Custom Parser Toolkit, the util directory has useful conversion programs and shell scripts, the client-toolkit directory contains the source code and documentation for the Z39.50 Client Toolkit (for both versions of Z39.50), the man directory has the UNIX-formatted manual pages, the sample directory has sample databases to experiment with, and the waisgate directory contains the waisgate utility, a World Wide Web-to-WAIS gateway for WAISserver.

This completes the installation of the WAIS software.

## Building an Example WAIS Database

Now you are ready to build a WAIS database. A WAIS database is made up of a set of data and a WAIS index. The installation includes some sample sets of data from which you can try building and searching

databases. Let's move into the sample directory and see what it contains.

```
% cd current/sample
% ls
Makefile  README  currency-exchange  data
% ls data
jargon  mail  data  weather
```

The /wais/current/sample/data directory contains several small sample data sets for you to practice with. Let's now create a WAIS database using the data in the resumes directory. First, let's take a look at our data set.

```
% ls data/resumes
README  res1.txt  res2.txt  res3.txt
```

The resumes directory contains three resume files, where the first line of each file contains the name. You may want to use your editor or the UNIX **more** command to view the contents of the files.

```
% more data/resumes/res1.txt
```

Now that you have a sample set of data, the next step is to build the WAIS index. If this is the first time you are doing this, you will need to create an index directory:

```
% cd /wais/current/sample
% mkdir indexes
```

A WAIS index is created from your original set of data using the **waisparse** and **waisindex** programs. The **waisparse** program takes a set of documents in the form of files and directories, and separates the data into a stream of documents. For each document, the parser identifies a headline and the content-bearing words. The **waisindex** program takes a stream of documents from **waisparse** and creates an index for fast search and retrieval of documents.

The command line for building an index of the resumes data is:

```
% waisparse -parse first-line
        /wais/current/sample/data/resumes/*.txt |
        waisindex -d /wais/current/sample/indexes/resumes
```

(This must be typed as a single line on the command line, without any carriage returns) The method of parsing is specified by the word after the -parse switch. This tells the parser that each file contains one document and that the headline is the first line in each file. Notice that a unix pipe is used to feed the output of **waisparse** into **waisindex**. The **waisindex** program reads the data from **waisparse** and builds a database, whose name is specified by the argument following **-d**

(/wais/current/sample/indexes/resumes in this example). The output you receive should look like the following:

```
17500: 0: Oct 18 18:36:54 1994: 7: parsing /wais/sample/data/resumes/res1.txt
17501: 0: Oct 18 18:36:54 1994: 100: indexing /wais/sample/data/resumes/res1.txt
17500: 1: Oct 18 18:36:54 1994: 7: parsing /wais/sample/data/resumes/res2.txt
17501: 1: Oct 18 18:36:54 1994: 100: indexing /wais/sample/data/resumes/res2.txt
17500: 2: Oct 18 18:36:54 1994: 7: parsing /wais/sample/data/resumes/res3.txt
17501: 2: Oct 18 18:36:54 1994: 100: indexing /wais/sample/data/resumes/res3.txt
17501: 3: Oct 18 18:36:54 1994: 100: flushing 1309 words (926 different words) to disk
17501: 4: Oct 18 18:36:54 1994: 100: generating headers
17501: 5: Oct 18 18:36:55 1994: 100: wrote source description /wais/sample/indexes/ resumes/resumes.src, please
examine it before use
17501: 6: Oct 18 18:36:55 1994: 100: building catalog
```

This completes the building of your WAIS database.

You've now built your first WAIS database. Let's check the results, and then run a query against it. The **waislookup** program provides an easy way to check that a database is built properly. The **waislookup** program can either search a WAIS database on the local file system, without talking to a WAIS server over the network, or it can connect to remote WAIS servers just like any other WAIS client. The **waislookup** program takes the name of the database following the **-d** switch. A sample interactive search session where **waislookup** searches a local database follows.

```
% /wais/current/bin/waislookup -d /wais/current/sample/indexes/resumes
local WAIS search on indexes/resumes
type: q          to quit
      r <N>      to retrieve document <N>
      f <N>      to feedback document <N>
      h <N>      to set maximum number of headlines to <N> (-1 = all)
      anything else performs a search
> unix AND programming
 0 score 1000 len   4102  John William Emmerson
 1 score    1 len    784 Query Report for this Search
> r 0
17527: 1: Oct 18 18:59:26 1994: 5: retrieving docID: "0 -4102 /wais/sample/data/resumes/res3.txt"
bytes 0 4102 from database "resumes" type "TEXT"
                  John William Emmerson
                    4126 Camry Avenue
...
> q
%
```

The **waislookup** program puts you into an interactive session in which you can ask questions of the information stored in the database. In this example, the first interactive command typed at the prompt is a question made up of the words "UNIX" and "programming" separated by the boolean operator "AND". This tells the **waislookup** program to look for documents containing both the words "UNIX" and "programming". **waislookup** returns the headline of one document (numbered 0) that

satisfied this criteria. The second interactive command is a retrieval request for the document numbered 0. **waislookup** returns and prints the contents of the document. To return to the UNIX prompt, use the **q** command.

Try running the sample **waislookup** session yourself to verify that your resumes database has been set up properly. Once you have completed this step, you are now ready to set up a server process.

## Setting up a WAIS Server

The WAIS server handles search and retrieval requests from clients. The server program is called **waisserver**. The remainder of this section demonstrates the usage of the **waisserver** program. The command line looks like:

```
% waisserver -p 8000 -d /wais/current/sample/indexes &
17561: 1: Oct 18 19:04:20 1994: 100: running WAIS Server 2.0.0
```

The **-p** switch gives the TCP port number that the server should use to communicate with client processes. The **-d** switch tells the server to find its databases in /wais/current/sample/indexes. The **&** at the end of the command line tells the shell to run the command in the background. This allows you to type other commands at the prompt while the server process is running.

Once the server process has been started, try running a search from a client. You may want to first try running the **waislookup** client on the same machine as the server, and then try it from a remote machine. (The **waislookup** program can search remote servers as well as local databases -- see below) If possible, run it from a different shell process, since the server prints out the record of each client/server transaction. A sample **waislookup** session on the resumes database follows (note: in the next command, the string "your.waisserver.host" should be replaced by the name of the machine on which your currently running the waisserver program)

```
% waislookup -d resumes@your.waisserver.host:8000
local WAIS search on indexes/resumes
type: q          to quit
      r <N>      to retrieve document <N>
      f <N>      to feedback document <N>
      h <N>      to set maximum number of headlines to <N> (-1 = all)
      anything else performs a search
> unix AND programming
17614: 1: Oct 18 19:07:12 1994: 100: initing connection to resumes@your.waisserver.host:8000
 0 score 1000 len   4102  John William Emmerson
 1 score   1 len    784 Query Report for this Search
> r 0
17527: 1: Oct 18 18:59:26 1994: 5: retrieving docID: "0 -4102 /wais/sample/data/resumes/res3.txt"
bytes 0 4102 from database "resumes" type "TEXT"
                    John William Emmerson
                     4126 Camry Avenue
                       ...
> q
```

Besides **waislookup**, other client programs are available. Several are included with your software installation, and several more are available by anonymous **ftp** from wais.com. See Appendix D for more information on client programs.

When the server responds to requests from a client process, the server prints out information to the shell. Your server output from processing the above **waislookup** session should look like:

```
17602: 2: Oct 18 19:07:12 1994: 100: child PID = 17615
17615: 0: Oct 18 19:07:12 1994: 1: accepted connection from: sushi [192.216.46.2], 2.0.0
17615: 1: Oct 18 19:07:13 1994: 100: init message: user <none> password <none> client WAIS Lookup
17615: 2: Oct 18 19:07:13 1994: 3: search database: "resumes" seed words: "unix AND programming"
17615: 3: Oct 18 19:07:13 1994: 100: checking access for client 192.216.46.2 to database
/wais/sample/indexes/resumes/index.acc
17615: 4: Oct 18 19:07:13 1994: 4: found 2 hits
17615: 5: Oct 18 19:07:17 1994: 100: checking access for client 192.216.46.2 to database
/wais/sample/indexes/resumes/index.acc
17615: 6: Oct 18 19:07:17 1994: 5: retrieving docID: "0 -4102 /wais/sample/data/resumes/res3.txt" bytes 0 4102
from database "resumes" type "TEXT"
17615: 7: Oct 18 19:07:20 1994: 2: closing connection;
17615: 8: Oct 18 19:07:20 1994: 2: done handling client
```

To kill the server process, first determine the process number (PID), and use the UNIX **kill** command.

```
% ps -auxww | grep waisserver
margaret   17561  0.0  2.2 2.16M  360K p4 S    0:00 waisserver -p...
margaret   17570  0.0  1.3 1.52M  208K p4 S    0:00 grep waisserver
% kill 17561
49  Terminated waisserver -p 8000 -d /wais/current/sample/indexes
```

After you've built your real WAIS database, you'll probably want to run your server in daemon mode under inetd as described in Chapter 5.

Now that you've experimented with a sample WAIS database, you can set up your real WAIS database. We recommend that you read through chapters 4, 5, and 6 to familiarize yourself with all the optional features you may want to activate. Nonetheless, you can simply return to the Quick Start section entitled "Building an Example WAIS Database", and follow the directions there, substituting your real data for the sample resume data.

# B

# Recommended Reading

The books listed here are ones we have found helpful. Most of them involve UNIX system administration.

Frisch, AEleen: *Essential System Administration*, O'Reilly & Associates, Inc., Sebastopol, CA 1991. Covers UNIX on a variety of platforms.

Nemeth, Evi, Garth Snyder, and Scott Seebass: *UNIX System Administration Handbook*, Prentice Hall Software Series, Englewood Clifs, NJ 1989. Good general UNIX system administration background, a bit outdated.

Stallman, Richard: *GNU Emacs Manual*, 6th ed., Freeware Software Foundation, Cambridge, MA, 1987. Emacs is our preferred UNIX editor. It's freely available, and much more powerful than vi, but also harder to learn.

Winsor, Janice: *Solaris System Administrator's Guide,* Ziff-Davis Press, Emeryville, CA 1993. If you are administering a Solaris machine, this book is absolutely essential.

# C

# Default Stopword List

The following words make up the default stopword list used by **waisparse**. They are included with your release in the file /wais/current/util/stopwords.txt. Stopwords are words which occur so frequently that they are not useful for distinguishing one document from another. Since they aren't useful for searching, they are not indexed.

a
about
above
according
across
actually
adj
after
afterwards
again
against
all
almost
alone
along
already
also
although
always
among
amongst
an
and
another
any
anyhow
anyone
anything
anywhere
are
aren
around
as
at
be
became
because
become
becomes
becoming
been
before
beforehand
begin
beginning
behind
being
below
beside
besides
between
beyond
billion
both
but
by
can
cannot
caption
co
could
couldn
did
didn
do
does
doesn
don
down
during
each
eg
eight
eighty
either
else
elsewhere
end
ending
enough
etc
even
ever
every
everyone
everything
everywhere
except
few
fifty
first
five
for
former
formerly
forty
found
four
from
further
had
has

hasn
have
haven
he
hence
her
here
hereafter
hereby
herein
hereupon
hers
herself
him
himself
his
how
however
hundred
ie
if
in
inc.
indeed
instead
into
is
isn
it
its
itself
last
later
latter
latterly
least
less
let
like
likely
ll
ltd
made
make
makes
many
maybe
me
meantime
meanwhile
might
million
miss
more
moreover
most
mostly
mr
mrs
much
must
my
myself
namely
neither
never
nevertheless
next
nine
ninety
no
nobody
none
nonetheless
noone
nor
not
nothing
now
nowhere
of
off
often
on
once
one
only
onto
or
other
others
otherwise
our
ours
ourselves
out
over
overall
own
per
perhaps
rather
re
recent
recently
same
seem
seemed
seeming
seems
seven
seventy
several
she
should
shouldn
since
six
sixty
so
some
somehow
someone
something
sometime
sometimes
somewhere
still
stop
such
taking
ten
than
that
the
their
them
themselves
then
thence
there
thereafter
thereby
therefore
therein
thereupon
these
they
thirty
this
those
though
thousand
three
through
throughout
thru
thus
to

together
too
toward
towards
trillion
twenty
two
under
unless
unlike
unlikely
until
up
upon
us
used
using
ve
very
via
was
wasn
we
we
well
were
weren
what
whatever
when
whence
whenever
where
whereafter
whereas
whereby
wherein
whereupon
wherever
whether
which
while
whither
who
whoever
whole
whom
whomever
whose
why
will
with
within
without
won
would
wouldn
yes
yet
you
your
yours
yourself
yourselves

# D

# WAIS Freeware Information

Freeware is, as it sounds, available for free. It is written and donated by individuals and therefore comes without any binding obligations to update, fix, or support it. Due to limited staff resources, WAIS Inc. cannot provide support for the freeware WAIS clients and servers.

For information on WAIS server freeware or the Clearinghouse of Networked Information Discovery and Retrieval (CNIDR), contact Jane Smith at jane.smith@cnidr.org or 919-248-9213. The director of the freeware is George Brett at ghb@jazz.concert.net or 919-962-1000. For information on WAIS clients, contact the individual developers of each respective client.

## WAIS Freeware Servers

| Server | FTP Location |
|---|---|
| NeXT | /pub/freeware/next/*@ftp.wais.com |
| RS6000 | /pub/freeware/rs6000/*@ftp.wais.com |
| SGI | /pub/freeware/sgi/*@ftp.wais.com |
| Source Code | /pub/freeware/unix-src/wais-8-*.tar.Z@ftp.wais.com |
| SUN | /pub/freeware/sun/*@ftp.wais.com |

## WAIS Freeware Clients

| Client | Author | FTP Location |
|---|---|---|
| DOS | Jim Fullton, UNC | /pub/wais/DOS/*@sunsite.unc.edu, or /pub/tcpip/pcwais.zip@hilbert.wharton.upenn.edu |
| GWAIS (Gnu Emacs) | Jonathon Goldman Thinking Machines Corp. | /pub/freeware/unix-src/wais-8-*.tar.Z@ftp.wais.com |
| IBM Mainframe | Tim Gauslin, USGS | /pub/freeware/ibm-mvs/*@ftp.wais.com |
| Mac | MCC Francois Schiettecatte | /pub/freeware/mac/MacWAIS*@ftp.wais.com /pub/freeware/mac/WAISBrowser*@ftp.wais.com |
| Mac HyperCard | Francois Schiettecatte | /pub/freeware/mac/HyperWais*@ftp.wais.com /pub/freeware/mac/JFIFBrowser*@ftp.wais.com |
| Mail | Jonathon Goldman Thinking Machines Corp. | send message to waismail@quake.think.com, "search <source-name> {keywords}" or "retrieve DOCID" (DOCID as returned by a search) |
| NeXT | Paul Burchard, Univ of Utah | /pub/freeware/next/*@ftp.wais.com |
| Openlook | Simon Spero, UNC | /pub/freeware/open-look/*@ftp.wais.com |
| OS2 | Kevin Oliveau, WAIS Inc Julie Mills, Library of Congress | /pub/freeware/os2/*@ftp.wais.com |
| SunView | | /pub/wais/sunview/*@sunsite.unc.edu |
| SWAIS | John Curran, BBN | /pub/freeware/unix-src/wais-8-*.tar.Z@ftp.wais.com |
| Telnet Access | (uses SWAIS) | Telnet wais.com, login wais, password user@host |
| VMS | Jim Fullton, UNC | /pub/wais/vms/*@sunsite.unc.edu |
| Windows | Tim Gauslin, USGS MCC | /pub/freeware/windows/wnwais*.zip@ftp.wais.com /pub/freeware/windows/EIWAIS*@ftp.wais.com |
| XWAIS | Jonathan Goldman Thinking Machines Corp. | /pub/freeware/unix-src/wais-8-*.tar.Z@ftp.wais.com |

# E

# Glossary of WAIS Terms

**.acc**

File extension for access files. An access file contains the IP addresses of all machines that are allowed to search the database. It is created by the database administrator to control access to a database.

**.cat**

File extension for catalog files. See also catalog.

**.dct**

File extension for dictionary files. A dictionary file contains the dictionary of all the words used in a database.

**.doc**

File extension for document table files. A document table file contains a record of each document in the database.

**.fn**

File extension for filename table files. A filename table file lists the filenames of the original data files.

**.hl**

File extension for headline table files. A headline table file contains the headlines of all the documents in the database.

**.inv**

File extension for inverted files. An inverted file lists of all the words in the database, and for each word all the documents which contain that word.

**.qst**

File extension used by question files. See also question structure.

**fielded search**

Performing a restricted search based on the value of field or set of fields is called fielded search.

**filter program**

A custom software component that retrieves and modifies data as an enhancement to the waisserver program. Filter programs are most often used to convert display formats and to process data from external databases.

**gwais**

**gwais** is a WAIS freeware client program for GNU-Emacs developed by Jonathan Goldman at Thinking Machines (jonathan@think.com).

**help**

If a WAIS client user types **help** as a query, or issues an empty query, one or more text files are displayed. If the file **index.hlp** exists in the WAIS database index directory, it is displayed. Otherwise, the **.src** file is displayed (unless the System Administrator has invoked **waisindex** with the **-nocat** option). In addition to one or the other of these, the **.cat** file is also displayed(unless the System Administrator has invoked **waisindex** with the **-nosrc** option).

**inetd daemon**

The **inetd** daemon is the Internet network daemon. This is a UNIX program that manages the network services according to the configuration specified by the **/etc/inetd.conf** file. The **inetd** daemon runs in the background and manages network requests by spawning off other daemons, or programs, to service these requests.

**multi-database**

A multi-database is a WAIS database composed of two or more subsidiary WAIS databases. It is easily defined by making an entry in the WAIS configuration file, index.wc.

**parse format**

A parse format determines how **waisparse** breaks a data file into its component documents, and decides which words to use as a headline. See also display format.

**phrase matching**

If the user's question contains a natural language expression made up of more than one word, called a phrase, a phrase-matching technique is employed. Using this technique, a higher weight is assigned to a document containing a phrase that identically matches the phrase in the user's question.

**plural stemming**

Plural stemming is a stemming algorithm that attempts to derive the root form of a word if given a plural.

**port**

A port number specifies a service on a UNIX machine. For example, WAIS usually runs on port 210, and Telnet usually runs on port 23. Ports below 1,000 are called "well-known" ports, and you must be superuser to run a program on them. The mapping between the service name and port number is determined in /etc/services file.

**porter stemming**

Porter stemming is a stemming algorithm that attempts to find the real base, or stem, of a word and derive any possible alternate variations.

**proximity relationship**

A proximity relationship designates that if the words in a question are located close together in a document, they are given a higher weight than those found further apart. The idea behind a proximity relationship is that words found in close proximity to each other in a document more likely contain the same content as that specified in the user's question.

**query**

See question.

**query report**

A query report is a document created by the server that describes how a client's question is parsed by the server.

**question structure**

A question structure is a file format used by some client programs to store the state of a user's inquiry so that it can be used again. It includes what databases to ask, the user's question, and any relevant documents. It may also store the last list of results.

**question**

A question is an expression containing a combination of natural language words and boolean operators.

**relevance feedback**

Relevance feedback is the ability to select a document or a portion of a document and find a set of documents "similar" to the selection. In essence, relevance feedback adds more words to the original question. It uses the most important words and phrases in the relevant document in addition to the original question. The most important words and phrases are determined by the same weighting algorithms as the words in the original question. The weight of the relevant document terms is less than the original question terms.

**relevance ranking**

Relevance ranking is the scoring of documents based on their relevance to a user's question, where the most relevant document has the highest score, or rank. A document receives a higher score if the words in the question are in the headline, or if the words appear many times, or if the phrases occur exactly as in the question. A document's score is derived from using techniques such as word weighting, term weighting, phrase matching, proximity relationships, and word density.

**results list**

The results list is the first thing returned to a client user in response to a query. It contains a list of headlines for the documents found along with the relevance rank or "score" assigned to each document. The client user may request retrieval of the listed documents, one at a time.

**right truncation**

A user can specify right truncation in a question by ending a word on the right with the wildcard character, '*'. This tells the WAIS server to search on words matching the base characters before the '*'. An example of using right truncation is a question such as "geo*", which may retrieve documents containing the words: geographer, geography, geologist, geometry, geometrical, etc.

**serve**

The transitive verb "to serve" is used when describing the services provided by a WAIS server. The server serves data to the client.

**server**

In a client/server architecture, the server is the program that provides the services requested by the client program. The machine on which the WAISserver software is installed is also know as the server.

**Solaris**

(aka. SunOS 5.x) Sun's version of AT&T's System V Release 4 UNIX operating system.

**source structure**

A source structure is a file format used by WAIS client programs to describe a particular database on a particular server. Typical contents are the machine name, ip address, port of the server, and the name of the database.

**standalone mode**

A WAIS server is run in standalone mode when the **waisserver** program is invoked directly by the user or shell script. See also daemon mode.

**stemming**

Stemming is a technique used to automatically derive the root of a queried word. The root is then used to search against the roots of the words contained in a database. If a question contains the word "skate", for example, stemming is used to find documents that may also include "skated" and "skating".

**stopword**

A stopword is a word that occurs so frequently that it is not useful for distinguishing one document from another. Since it is not useful for searching, it is not indexed.

**superuser**

On machines running the UNIX operating system, the superuser is a privileged user who has access to all files and all services provided by the machine. This status is usually reserved for system administrators.

**swais**

**swais** is a freeware WAIS client program for ASCII terminals developed by John Curran.

**TCP/IP**

TCP/IP is an acronym for Transmission Control Protocol/Internet Protocol. This is the low level protocol which is used on the Internet, and on many LANs. It provides reliable data communication.

**term weight**

Each word used in a database is assigned a numerical value, called the term weight, based on the frequency of occurrence of that word over all documents in the database. Words that occur frequently throughout the database are not weighted as highly as the words that appear less frequently. Very common words are either ignored or diminished in the scoring. For example, since the term, "animal", may occur frequently in many of the documents in a database, its term weighting is small compared to a term such as "hippopotamus", which may occur in only a small number of documents.

**WAIS**

WAIS is an acronym for Wide Area Information Servers and a trademark of WAIS Inc.

**WAIS database**

A WAIS database consists of a set of documents, a set of index files, and a source file. See also database.

**WAISgate**

The WAISgate product is a gateway between WAIS and the World-Wide Web, an Internet browsing and retrieval tool based on hypertext links between information sources.

**WAIS protocol**

The WAIS protocol is used to connect WAIS clients and servers. It is based on the Z39.50 protocol. Because a standard protocol is used, clients and servers can be built on a wide variety of computer architectures communicating over local and wide-area networks.

**waisindex**

This program takes the documents specified by waisparse and builds a WAIS searchable index.

**waislookup**

This program is essentially a very simple WAIS client generally used for testing a WAIS installation. It provides an interactive, dumb terminal interface. It can be invoked as an interactive server, in which case it does not go through the protocol or server but rather does the search internally. This mode is useful for troubleshooting WAIS indexes. The waislookup program can also be invoked as a client and then communicate with either a local or remote WAIS server.This mode is useful for troubleshooting WAIS servers.

**waisparse**

This program reads a data file and breaks it into its component documents, decides which words to index, and which words to use as the headline. The output of waisparse is fed to waisindex.

**waisreporter**

this program summarizes the log files generated by waisserver. It can be used to monitor database usage.

**WAISserver**

The WAISserver product is software licensed and sold by WAIS Inc.. Properly installed, it enables a computer to act as a WAIS server. Included in the WAISserver software are the waisserver, waisindex, waisparse, waislookup, waisdelete, and waisreporter programs.

**WAIS server**

The term "WAIS server" is used to describe any software or hardware that serves one or more WAIS databases by using the Z39.50 communication protocol.

**waisserver**

This program waits for a connection from a client, and handles the connection by searching and/or retrieving documents from a WAIS database.

**word density**

The ratio of the number of times a queried word appears in a document to the size of the document is called the word density.

It is a measure of how important a queried word is to the overall content of the document. A higher word density results in a higher relevance ranking.

**word weight**

If a word in a document is found to match a word in the user's question, the word is assigned a word weight, and this weight additively contributes to the overall score of the document. The exact weight that a word receives depends on where in the document the word was found. A word is weighted highest if it appears in the headline, less if the word appears in all capital letters or if the first letter of the word is capitalized, and finally, a word has the least weight if it appears only in the text.

**xwais**

**xwais** is a freeware WAIS client program for X-Windows developed by Jonathan Goldman at Thinking Machines (jonathan@think.com).

**Z39.50**

Z39.50 is the National Information Standards Organization (NISO) protocol for information search and retrieval.

# Index